The University of Jordan
Authorization Form

I, Mahran Meti'b Al-Zyoud authorize the University of Jordan to supply copies of my Thesis/Dissertation to libraries or establishments or individuals on request.

Signature: [signature]
Date: 23.10.2011

الجامعة الأردنيـــة
كلية الدراسات العليا

التاريــــخ: / /

نموذج رقم (١٨)
اقرار والتزام بقوانين الجامعة الأردنية وأنظمتها وتعليماتها لطلبة الماجستير

أنا الطالب: عمران منتصب محمد الزيود الرقم الجامعي: ٨٠٩٠٤٩٩
التخصص: أنظمة المعلومات الحاسوبية الكلية: الملك عبدالله الثاني لتكنولوجيا المعلومات

اعلن بأنني قد التزمت بقوانين الجامعة الأردنية وأنظمتها وتعليماتها وقراراتها السارية المفعول المتعلقة باعداد رسائل الماجستير والدكتوراة عندما قمت شخصيا" باعداد رسالتي / اطروحتي بعنوان: ....
Towards a Model of Context Awareness Using Web Services
....

وذلك بما ينسجم مع الأمانة العلمية المتعارف عليها في كتابة الرسائل والأطاريح العلمية. كما أنني أعلن بأن رسالتي /اطروحتي هذه غير منقولة أو مستلة من رسائل أو أطاريح أو كتب أو أبحاث أو أي منشورات علمية تم نشرها أو تخزينها في أي وسيلة اعلامية، وتأسيسا" على ما تقدم فانني أتحمل المسؤولية بأنواعها كافة فيما لو تبين غير ذلك بما فيه حق مجلس العمداء في الجامعة الأردنية بالغاء قرار منحي الدرجة العلمية التي حصلت عليها وسحب شهادة التخرج مني بعد صدورها دون أن يكون لي أي حق في التظلم أو الاعتراض أو الطعن بأي صورة كانت في القرار الصادر عن مجلس العمداء بهذا الصدد.

توقيع الطالب: .... التاريخ: ٢٨ / ١٠ / ٢٠١١

تعتمد كلية الدراسات العليا
هذه النسخة من الرسالـــة
التوقيع.......... التاريخ ٢٣ / ١١ / ٢٠١١

TOWARDS A MODEL OF CONTEXT AWARENESS USING WEB SERVICES

By

Mahran Meti'b Al-Zyoud

Supervisor

Dr. Nadim Obeid, Prof.

This Thesis was Submitted in Partial Fulfillment of the Requirements for the Master's Degree of Science in Computer Information Systems

Faculty of Graduate Studies

The University of Jordan

October, 2011

COMMITTEE DECISION

This Thesis/Dissertation (Towards a Model of Context Awareness Using Web Services) was Successfully Defended and Approved on 10/10/2011

Examination Committee	**Signature**
Dr. Nadim Ali Meri Obeid (Supervisor) Prof. of Artificial Intelligence	------------------------------
Dr. Fawaz Ahmad Al-Zaghoul. (Member) Prof. of Quality Metrics in Software Engineering	------------------------------
Dr. Moh'd Belal Al-Zoubi (Member) Prof. of Data Mining / GIS and Computer Graphics	------------------------------
Dr. Hamed Saqer Al-Bdour (Member) Assoc. Prof. of Computer Systems and Network (Mu'tah University)	------------------------------

DEDICATION

To my parents who always emphasize the importance of education. Thanks a lot for your love and encouragement throughout my life. Wish you a life full of health and peace.

AKNOWLEDGEMENT

In the name of Allah who is the most gracious and merciful. We are thankful to our creator who blessed us with abilities to overcome the difficulties of life.

First of all, I would like to thank Dr. Nadim Obeid who has been the ideal thesis supervisor. His sage advices, insightful criticism, and patient encouragement aided me in the writing of this thesis. I would like to thank this kind person for giving me the morale and the knowledge needed to go through the difficulties faced while accomplishing the task.

In addition, I would like to thank every instructor in the King Abdullah II School for Information Technology.

I am also grateful to the examination committee.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ATC	:	Air Traffic Control
API	:	Application Programming Interface
CBR	:	Case-Based Reasoning
CSCP	:	Comprehensive Structured Context Profiles
FAA	:	Federal Aviation Administration
FDAU	:	Flight Data Acquisition Unit
FDR	:	Flight Data Recorder
FTP	:	File Transfer Protocol
GPS	:	Global Positioning System
HTTP	:	Hyper Text Transfer Protocol
IP	:	Internet Protocol
IS	:	Information System
JDBC	:	Java Database Connectivity
MVC	:	Model-View-Controller
OOP	:	Object Oriented Paradigm
ORDB	:	Object Relational Database
ORM	:	Object-Role Modeling
PC	:	Personal Computer
PDA	:	Personal Digital Assistant
PL/SQL	:	Procedural Language/Structured Query Language
SOA	:	Service Oriented Architecture
SOAP	:	Simple Object Access Protocol
SoC	:	Separation of Concerns
SMTP	:	Simple Mail Transfer Protocol
SQL	:	Structured Query Language

UDDI	:	Universal Description, Discovery and Integration
UML	:	Unified Modeling Language
UoJ	:	University of Jordan
WSDL	:	Web Services Description Language
XML	:	eXtensible Markup Language

TOWARDS A MODEL OF CONTEXT AWARENESS USING WEB SERVICES

By
Mahran Meti'b Al-Zyoud

Supervisor
Dr. Nadim Obeid, Prof.

ABSTRACT

Recent years have witnessed the movement of many applications from the traditional closed environments into open ones particularly the Internet. These applications, which are being accessed via web browsers, usually offer a great amount of information and services. These two facts, open environments and content explosion, constitute a true hindrance on the usability of these web applications, where usability measures the degree of usage satisfaction of the application provider and the application user.

The usability of web applications could be improved if the application knows the current user's context in order to adapt its offerings accordingly. Context has the potential to make difference, and both sides of communication (the web application and the device accessing it) must work together to bring the benefits of context awareness.

In an attempt to increase the usability of web applications, this thesis focuses on modeling context awareness. We propose two layered models that complete each other and organize the knowledge in layers in order to give the web applications' developers the adequate knowledge and a visualization of what should be done to develop a context aware application.

Two major issues in the implementation are context representation and the heterogeneity that characterizes the open environment of web applications. Context representation, in our work, follows the object oriented approach, and the web services were utilized to create the universal interoperability which should facilitate the communication between the server hosting the web application and the devices used to access it.

Two case studies selected from two distant domains were given: a prototype of University of Jordan Portal was implemented, and a Decision Support System deployed in Airplanes was discussed according to our vision of a possible implementation but due to time limitation it still needs more elaboration.

CHAPTER ONE

INTRODUCTION

1. INTRODUCTION

Businesses and institutions create web applications that aim to serve as an entrance to their information systems or to announce their existence to the public. These web applications usually include huge amounts of important information and distinguished services that are targeted to users according to their preferences, interests, roles in the business, geographical location, time of access, etc. Based on that we, in this thesis, propose a way to make these web applications more and more usable and achieve the intended goals of their presence.

1.1 Problem Overview

The way to enhance the usability of web applications is to make them adaptable. But adaptation needs a rigid ground to be based on. In this work, we suggest that this ground should be the context.

This suggestion brings a set of questions and difficulties of how to collect, store, analyze, disseminate, and query context in order to adapt. Some of these concerns (collecting and analyzing) are out of the scope of this thesis.

The existence of web applications in open environments such as the Internet put the heterogeneity issue in front of the web applications' developers who would like to utilize context appropriately. We aim at making each device responsible for its own context without the need of the web application to know the details of how the device is managing the context, but there must exist a common ground to reliably exchange the messages between the web application and the device without error or misunderstanding; i.e. to make them interoperable regardless of hardware architecture or software platform differences.

1.2 The Significance of the Study

In our opinion, the importance of how and what to present equals the content itself. Hence, web application's importance emerges from its content and how this content is being presented to the intended user.

This thesis emphasizes on the major role of context in achieving a high level of adaptation to the web application in order to solve the problem of content explosion.

At the end, every user wants to be served at the right time in exactly the way he/she wants. The web application, that an employee uses to access an information system or the ordinary navigator uses to buy, entertain, or learn, should know about the user what the user himself/herself wants it to know and eventually the application should try to help him accomplish the mission in a reasonable time. We, in this thesis, worked to achieve that by putting the knowledge needed in layered models and implementing them using those technologies available at the moment. We used the role, time, location, and device as the contextual aspects of interest.

1.3 Research Objectives

The main goals of this work are:

1. Creating a layered model of context awareness
2. Implementing the model suggested in a way that achieves the requirements put in the model especially the interoperability requirement.

1.4 Thesis Organization

The thesis is organized as follows: chapter two gives a literature review. An overview of context, context awareness, and representation of context data is presented. Chapter three presents some background on Web Applications, Web Services, the design process of context-aware applications, difficulties facing that process, in addition to a discussion on databases with concentration on the object orientation features supported. Chapter four introduces the models, in addition to the ideas and concepts that lead us to the models suggested. Chapter five is dedicated to the implementation of the models and at the end of the chapter; scenarios of usage in two separate domains are discussed. Finally, in chapter six we present the conclusion of the research along with future works and suggested improvements.

CHAPTER TWO

LITERATURE REVIEW

2. Literature Review

In this chapter we present how context is viewed and defined by researchers in different fields especially in computer science, and what context aware applications have been suggested or implemented. In addition, we see what adaptation could bring to a web application, and what context modeling approaches are there with a concentration on the object oriented approach.

2.1 The Notion of Context

Different research areas have studied context and as a consequence there are many definitions of the term according to each field's research interests.

In this section, we look at some of these definitions to get a closer idea about what context is.

We list some of context definitions presented in the literature and mainly those discussed by Bradley and Dunlop (2005) who investigated what context meant to researchers in three domains that are considered to be the most applicable to and beneficial for context-aware computing, and those are: linguistics, computer science, and psychology.

- Webster's New World Dictionary (1970) gave the following definition of context: "the whole situation, background, or environment relevant to a particular event, personality, creation, etc."
- Compact Oxford English Dictionary (2005) defines context as: "the circumstances that form the setting for an event, statement, or idea".
- McCarthy (1993) defined context as "a generalization of a collection of assumptions".
- Br´ezillon (1999) defined context as "a focus of attention".

- In linguistics, Fetzer (1997) defined context as a tripartite system of objective, social and subjective worlds, their sub-systems and presuppositions. Whereas Bunt (1997) grouped the relevant factors of conversational context into five categories: Linguistic, Semantic, Physical, Social, and Cognitive.
- In the domain of computer science, the following definitions were given:
    - ❖ Dey (2001) has the most cited definition of context. He defined context as "any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves".
    - ❖ Ward, et al. (1997) defined context as "the state of the application's surroundings ".
    - ❖ Brown (1996) defined context as "the elements of the user's environment that the user's computer knows about"
    - ❖ Chen and Kotz (2000) considered context as "the set of environmental states and settings that either determines an application's behavior or in which an application event occurs and is interesting to the user".
    - ❖ Dey, et al. (1999) stated that context is "the user's physical, social, emotional, or informational state"
    - ❖ To Zetie (2002), context is "the knowledge about the goals, tasks, intentions, history and preferences of the user that a software application applies to optimizing the effectiveness of the application"
    - ❖ Schilit, et al. (1994) treated the user as the primary entity to ask questions such as where you are, who you are with, and what resources are nearby.

- Schmidt (2001) defined it as "facts that matter for an application/user/device, which are inherently connected to time and location"
- Funk and Miller (1997) defined it as "everything surrounding an item of interest, including the 'mindset' of any humans involved in the context"

2.2 Context Aware Applications

Context awareness may be defined as the ability of a computer system to know about the surrounding environment of the user (the context); i.e. to be aware of the user's context. This awareness can facilitate the job of the user and make him doing it more efficiently by minimizing the amount of explicit input from the side of the user; hence reducing the time required to accomplish tasks and utilize the time concentrating on the core job itself.

The following is a list of some types of context-aware applications implemented, partially or fully, and investigated in the literature:

- Jahnke et al. (2004) created a system that is aware of the patient's condition with a repository containing his/her condition's history in order to enhance the medical care provided.
- Schilit et al. (2003) described a kind of applications that serves as a friend finder by taking the user type and location as criteria to suggest and find friends. Facebook now supports this type of service.
- Louis and Shankar (2004) discussed how the presentation of different context sources would improve the user's interaction with calendar and appointment applications.

- Beigl et al. (2003) talked about how being aware of context could optimize and enhance the performance of some applications by predicting what the user is likely to do next and prepare whatever resources required accordingly. These resources may include smart caching in information retrieval.
- Lamming and Flynn (1994) discussed a very important application type serving as a memory aid that aims to improve information retrieval. It works by storing regular information along with contextual information pertaining to the information such as the location and the mission being done at the moment of registering the information.

2.3 Adaptation

Through various readings, we can conclude the following rule: To improve usability the web application should adapt, and thus should be context aware. So the adaptation is one of the goals of being context aware.

There are many aspects of the web application that can be adapted; among these we may cite the following primary aspects:

- Content: the web application can adapt what it offers to the end user. This includes the information and the services, i.e. what elements selected and what determines them. In the UoJ portal, which is a prototype implementation of the University of Jordan website shown in the implementation chapter, we'll see how the same page requested may display different content according to the user's role, location, or time of access. For example, if the user is accessing the page from inside UoJ, then the page may display advertisements of current

place of access; whereas if the access is from outside, then more general advertisements may appear.

- Presentation: the appearance of the application's user interface could be adapted according to the device's display size or the user movement (bigger font size, for example) as suggested by (Schmidt, 2000).
- Navigation: the navigation (links, for example) hierarchy can be adapted dynamically according to user's inferred goals or patterns of usage. Also in the implementation chapter, the page may suggest different set of link based on the user's role. For example, if the user is one of the faculty members then the links may suggest going to Acad (academic system used to enter students' grades) system or employees' email; whereas if he is a student then the set of links may suggest going to Reg (students' registration page) or students' email.

2.4 Context data modeling

An efficient model for handling and storing context data in a machine processable form is crucial for context aware systems (Baldauf et al., 2007).

In their survey, Strang and Linnhoff-Popien (2004), state that "a well designed model is a key accessor to the context in any context-aware system". This is one of the primary pillars and motives of this work.

The main context modeling approaches are summarized in the following section, whereas the section (2.4.2) gives more detailed presentation about the object oriented approach of modeling context, which is more relevant to our work in this thesis, by presenting projects adopting this approach.

2.4.1 Context Modeling Approaches

Approaches of modeling context according to the schema of data structures used to exchange contextual information are summarized in the table (2.4.1.1):

Table 2.4.1.1: Context Modeling Approaches

Approach	**Description**	**Projects (References to approach usage)**
Key-Value Modeling	Simply, can be though of as environment variables having different values. This modeling approach is easy to manage.	(Schilit et al., 1994)
Markup Scheme Modeling	Hierarchical data structure consisting of markup tags with attributes and content.	(Held et al., 2002) (Comprehensive Structured Context Profiles (CSCP))
Graphical Modeling	graphical components created by an instrument such as Unified Modeling Language (UML)	(Bauer, 2003) (modeling context pertaining to air traffic management) (Henricksen et al., 2003) (Object-Role Modeling (ORM))
Object Oriented Modeling	encapsulation and reusability are used to cover parts of the problems arising from the dynamics of the context in ubiquitous environments (read more in the next section)	(Schmidt et al., 1999). (Cues) (Schmidt and Laerhoven, 1998) (TEA project) (Cheverst, et al., 1999) (ActiveObject Model of the GUIDE project)
Logic Based Modeling	Formality is overwhelming in this approach. Here context is represented as facts, expressions and rules. Adding, updating, and deleting of contextual information are done in terms of facts or inferred from the rules.	(Gray and Salber, 2001) (Sensed Context Model) (Bacon et al., 1997) (the multimedia system)
Ontology Based Modeling	Ontologies describe information of some domain by specifying concepts and interrelations.	(Wang et al., 2004) (CONON) (Chen, et al., 2003) (CoBrA)

2.4.2 Object oriented approach of modeling context

Context modeling has seen an opportunity in using object orientation to represent contextual information. The advantages of this approach make it possible to hide (encapsulate) context processing on an object level. Only the output can be accessed through the determined interfaces. In this thesis we adopt this approach to be part of the model and the implementation where we utilize the capabilities of Object Relational Databases.

Here we present some of the applications where object oriented modeling is used.

- Cues

  Cues are abstractions from sensors, whether being physical or logical sensors.

  A Cue is defined as "a function taking the values of a single sensor up to a certain time as input and providing a symbolic or sub-symbolic output" (Schmidt et al., 1999).

  The reason why this project is considered to be an example of object oriented modeling is that cues, as their definition shows, are providing contextual information with each cue having a set of possible values defined as output with the consumer of these cues, who asks for context through the interfaces, having no idea what processing has been done to infer these outputted values (Encapsulation). Here context is derived as a description of the current situation on an abstract level from the cues that are available. Hence, cues are objects used to help infer the user's objective.

- Hydrogen

The figure (2.4.2.1) shows how Hydrogen (Hofer, et al., 2002) is considering context in an object oriented way of thinking.

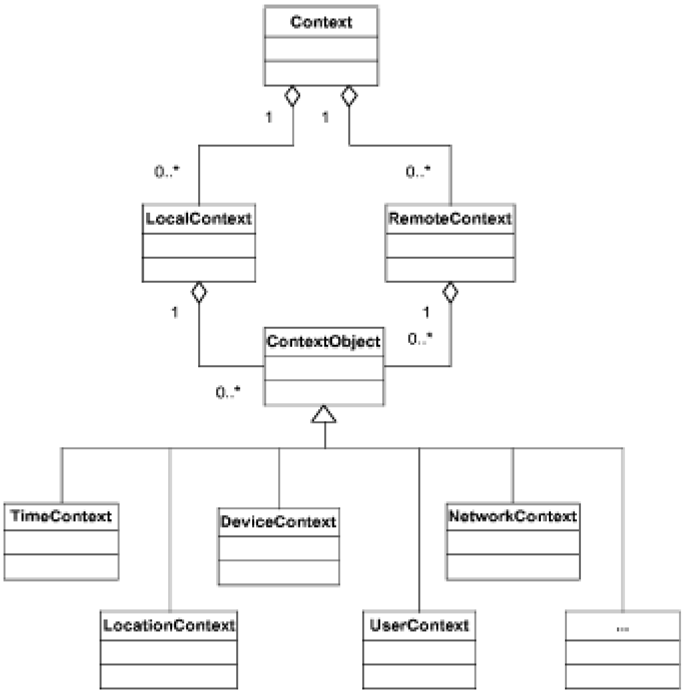


Figure 2.4.2.1: Hydrogen's object oriented approach

In figure (2.4.2.1), the device's context comprises the local context of its own and the remote context of other devices. These remote and local contexts are represented as ContextObjects. ContextObject is the father (the super class), while LocationContext, TimeContext, UserContext, and others are considered to be the children; those taking common attributes and behaviors from the father and adding their own attributes and behaviors to constitute specializations. The approach permits to add new context types by simply extending the father ContextObject. There are abstract methods existing in the

super class ContextObject which should be implemented by children objects to convert data streams from XML representations into ContextObjects and vise versa.

- GUIDE

The GUIDE project, which aims at providing a context sensitive tourist guide, is visualized as an object model (Cheverst et al., 1999). The goal of object modeling here is to be able to manage a great variety of personal and environmental contextual information while maintaining scalability. Being an object oriented approach of modeling; it encapsulates the details of data collection and fusing (e.g. the context adaptive composition of HTML fragments) within the active objects in an attempt to hide them to other components of the system. The figure (2.4.2.2) shows the design of the model.

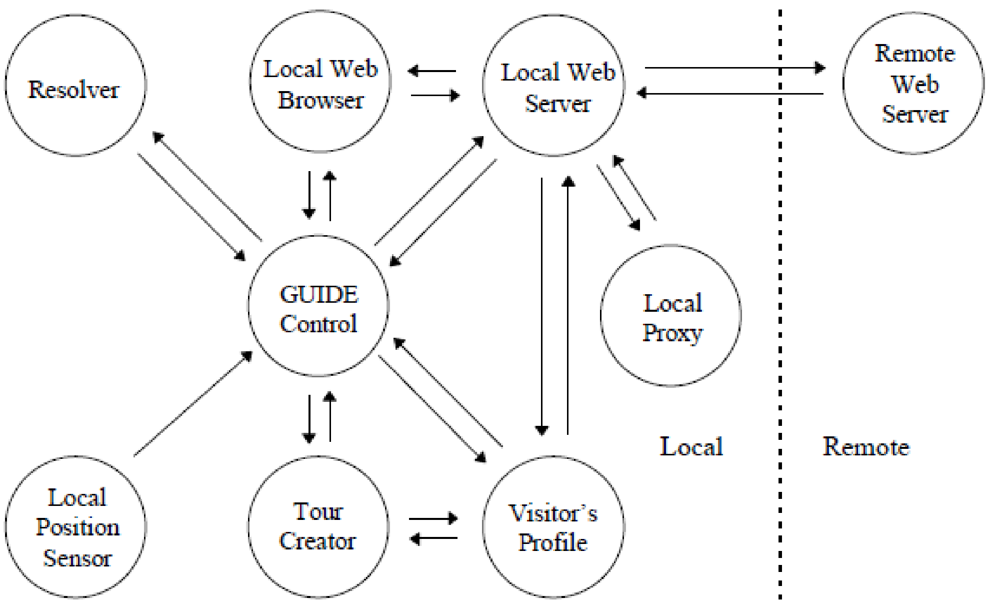


Figure 2.4.2.2: The GUIDE object model

One of the objects, the Visitor's Profile object, stores the visitor's current preferences. It provides a set of methods for returning the visitor's details. One of the functions of the Control object is to store the visitor's current location.

We found that some of the methods such as returnPreferredLanguage can be considered a hindrance that we must avoid in any suggestion we'll make in our modeling and implementation, since providing such an interface naming makes things less dynamic. We can't imagine that for every profile entry (context), for example, we have to provide its own method that says returnProfileEntry!! That would be cumbersome and lacks dynamicity.

- Go playing programs

Bouzy and Cazenave (1997) have used the object oriented paradigm to represent the contextual knowledge in an attempt to simplify the representation in complex domains. They proposed general object oriented mechanisms to represent context in complex systems. Object Oriented Paradigm (OOP) seems appropriate to represent context. They used systems that play the game of Go to introduce examples of what can be done by using OOP mechanisms. These examples included the following:

  - a general class with specialized classes,
  - a master slot of a class and slave slots,
  - a list of goals that depend on each other.

The mentioned game has its own set of rules, which have been explained in the paper, that when analyzed carefully will lead to the work represented in the figures they have produced. But to take a hint from what they did, see figure (2.4.2.3).

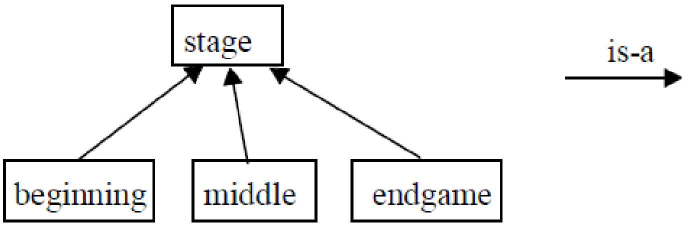


Figure 2.4.2.3: the game Go - a general class with specialized classes

Figure (2.4.2.3) shows how they tried to represent the temporal context which is just one part of the context surrounding the game. Other context types dealt with in the game are: spatial, goal, and global contexts. In the figure, each specialized class represents one specific temporal context. Each one of these classes has a set of meta-rules that permits the activation of specific rules' packets in order to generate the relevant kinds of moves in the global move decision process.

This means that when the slice of time labeled as middle has finished, the program will start to use those set of packets of rules advised by the "endgame" meta-rules.

In object oriented terminology, it is simply a general class that has been specialized into more specific classes and these classes have been created to deal with context.

The implementation of object orientation in all the previously mentioned projects and papers emphasizes its importance and feasibility to represent contextual data. We, in this thesis, believe that the application of objected oriented mechanisms offers a kind of well-organized outputs derived from a deep analysis. In addition, object orientation gives the capabilities needed to simulate what happens in reality to a great degree in a straight forward manner. Hence, we have applied it as much as possible throughout this thesis and in particular in context representation as will be discussed later.

2.4.3 Approaches to access contextual information.

This section gives an overview of some of the approaches proposed to access contextual information.

Chen, et al. (2004) presents three different approaches for contextual information acquisition: Direct sensor access, Context server, Middleware infrastructure. Direct sensor access tightly couples the contextual information acquisition and the logical processing. Context server approach gives the ability to access contextual information simultaneously from the server that collects all sensors' data and that has the responsibility of performing any computations needed on this data. The middleware-based approach is a layered architecture to context-aware systems that separates the business logic from the data acquisition and hence it hides the low-level sensing details. This approach increases the reusability of the sensing infrastructure compared to direct sensor access, and eases the extensibility due to the stability of the client code which has not to be modified anymore.

2.5 Context awareness models

This section gives an overview of the principal suggested models of context awareness.

Winograd (2000) discussed three models proposed by different researchers for coordinating multiple processes and components of context aware systems:

- Widgets. Widgets are software components that are created as a way to encapsulate the driver details of a device and other low level issues in order to create a uniform way of communication to provide contextual data directly to the applications. The interaction is implemented in terms of messages to widgets, and callbacks - when a certain event is registered by the sensor. In this approach, there is a widget manager that has the responsibility of coordination, and widgets are components within the manager process, rather than being implemented as independent processes. A drawback in this approach is that it is susceptible to component failures that might impact the normal functioning of the system.

- Networked services: this model utilizes the flexibility of client-server architecture where the client and server reside at different network locations and communicate using Internet protocols. This model is considered to be robust since it does depend on a central manager like the above mentioned model. The process goes as follows: the client needs to find the location of a service (through some kind of resource discovery for example), and then set up a connection with it. Connectivity is based on finding the network location

(host and port) of the service and providing software that uses the service's protocol for encoding content.

- Blackboard model: the above models follow a process-oriented approach, while the blackboard the data-centric view where processes send messages to a common board (e.g. context server) and subscribe to be notified with messages matching a specified pattern that have been posted. In this approach all communication must go through a centralized server. This model is considered the most loosely coupled and hence it pays a price in communication efficiency

CHAPTER THREE

RELATED BACKGROUND

In this chapter, we present a relevant background to our work. This includes: web applications and its generations, and web services and its dependencies. In addition, we investigate a bunch of issues confronting the mission of creating context-awareness architecture. We also review the process that must be undergone in order to design and build a context-aware application.

3.1 Web Applications

A web application is an application that gives access to an information system over the network (Intranet or Internet). These applications offer a variety of information and services that often need complex interaction.

Some of the advantages of web applications are (Valums, 2010):

- it only needs a web client (usually a browser).
- it usually requires no or less disk space on the client.
- it needs a simple deployment procedure (if it needs at all).
- no upgrade procedure for new features is needed.

According to Dumke et al. (2003) the following generations are the most prominent in the evolution of web applications:

- First generation: this stage is characterized by publishing information on the Internet, and the ability to view this information by any user with a Web browser.

- Second generation: in this stage the user interaction with some server components is possible. These components may provide complex information services, for example by integration with a database that made accessible through the Web.
- Third generation: here the creation of plug-ins for the display of special media types in the browser becomes available.
- Fourth generation: this generation is characterized by the ability to generate pages dynamically by the Web server. Note that the information provided by the system is managed consistently on the server.
- Fifth generation: this generation aims at moving into what is called a “Semantic Web” where association of meaning to information on the Web allows for smarter machine processing of this information.

Moving the application from the traditional controlled environment where a custom client has to be used for communication into a more open environment (Internet, for instance) decreases the likelihood for the service provider to know about the client. We, in this work, see the potential of adding context to the picture and work towards making web applications context-aware.

3.2 Web Services

The usage of web services as an essential part in the implementation of the suggested model in this thesis has its own reasons. In the following sub-sections we give an overview of the web services technology in order to reveal these reasons.

3.2.1 Why to use Web Services?

Web Services are loosely coupled software components that can be reached using open protocols. Two primary protocols are: Hyper Text Transfer Protocol (HTTP) and eXtensible Markup Language (XML). Web Services technology is an effort to create a universal interoperability by facilitating the communication between electronic devices on the network.

To elaborate the idea; consider few persons where each one speaks a different language than the other, comes from an environment with its own social habits and unique nature, and each one of these persons has useful information that the other wants to know. In an attempt to communicate with each other; they have agreed to use a kind of a common language that all will read, listen, and speak. This agreement on a common language becomes more beneficial when the number of persons increases. This is exactly what happens in Web Services; different devices with different hardware architectures, operating systems installed, information systems implemented by different programming languages and these devices want to speak with each other to take benefits of the information sources they have.

Web Services, basically, can be considered as a universal client/server architecture that allows systems to communicate with each other without having to use any proprietary client libraries. It converts various information sources into components that enhance Internet and Intranet applications whatever the size of the business is,

and that is exactly what we seek in this thesis: to enhance the web application by providing it with the context (information source) of the user.

3.2.2 Dependencies and parts of the technology

- Web Services Description Language (WSDL): to make the lines of code get executed on the information source side, the requestor should contact the responder (the web service) via a kind of an interface. The service interface information is disclosed to the client via a configuration file encoded in a standard format (WSDL). This language describes the web services by using XML grammar.

- Universal Description, Discovery and Integration (UDDI): there should be registries where different service providers who have web services can use to make notes of their services, and these registries should be platform-independent. The UDDI could be imagined to be as a phonebook containing yellow, white, and green pages that can be inquired to provide information about the web services' providers in a variety of ways.

- Simple Object Access Protocol (SOAP): this protocol is concerned with the messaging issue. SOAP acts as an envelope for XML-based messages that is used for exchanging structured information. Since there was a need to overcome the heterogeneity hindrance, SOAP was invented to transfer between the requester and the provider independently of any runtime environment.

- XML: XML is a mechanism for storing structural data in text files. Two characteristics of XML are interoperability and extensibility. Presentation of data in a document in such a hierarchical tree format that ultimately creates a presentation of structured data is the primary concern of XML (Hunter et al., 2007).

  As an important implementation note, Web Services technology uses XML on top of HTTP, and since the firewalls don't usually block the HTTP port, no problem is expected with them. The traffic could also be transferred using FTP or SMTP.

Figure (3.2.2.1) shows the main components of the Service Oriented Architecture (SOA) which is based on Web Services as depicted and discussed in (Michlmayr, et al., 2007).

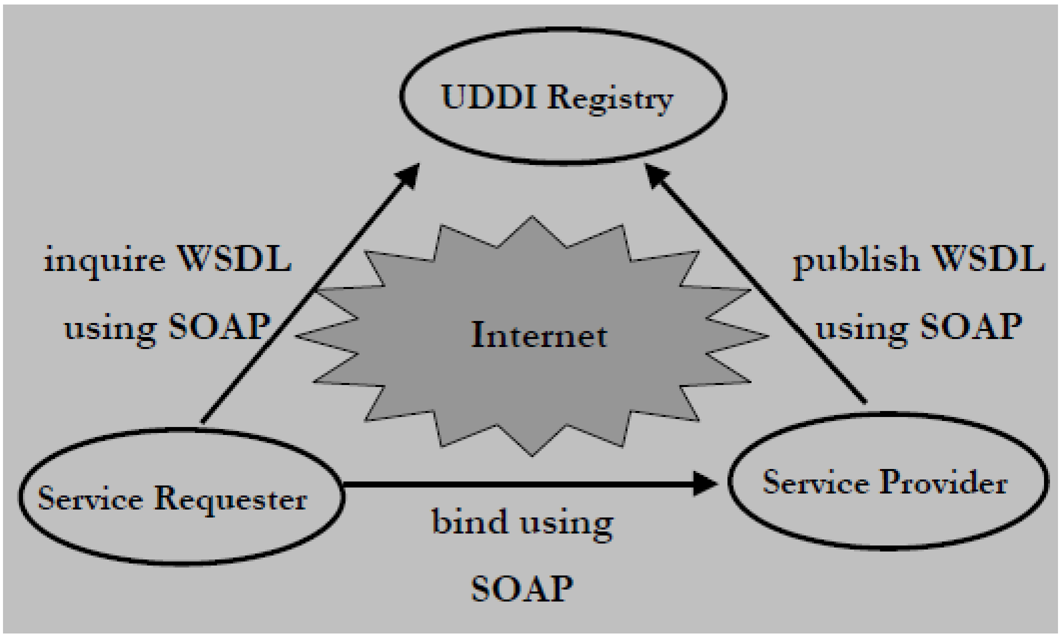


Figure 3.2.2.1: Service Oriented Architecture Triangle

- Service Broker: the broker sets in the middle between the requester and the provider and acts as a registry that answers the question of what services exist at where.
- Service Provider: the side that contains the functionality and wishes to offer it to the clients. It publishes this intent to the broker.
- Service Requester: the side that wishes to use a certain kind of functionality. It asks the broker where to find this type of functionality and then it binds itself to the web service at the provider having the functionality it needs.

What usually happens is as follows (Gunzer, 2002):

1. The Web Service's provider describes the Web Service in a WSDL document and publishes it to a UDDI registration using the Publisher's API (which is based on SOAP).
2. A service requester uses the UDDI Inquiry API to search the UDDI registration for a suitable service provider.
3. A SOAP request will be created according to the WSDL document.
4. The SOAP request will be sent to the service provider, and the response will be processed.

It is important to mention at this stage that what we are suggesting in this thesis is the elimination of the publishing (step 1) and discovery phase (step 2) and hence there will be no need for a broker. This could happen because we are dealing with the same web service's interface as the implementation will show. The main idea is as follows: why to discover something that I know. A kind of agreement should be there

regarding the interface signature, and the IP address of the node accessing the web application could be extracted from the request, so no discovery should be done.

3.3 Design Process of Context-Aware Applications

Abowd et al. (1998) discussed the process that must be undergone in order to design and build a context-aware application. The design process consists of the following stages:

- Specification: this stage aims at specifying two main concerns:
    1. The behaviors that should be implemented
    2. The contextual information that must be present at the running time of the application in order to execute the behaviors

- Acquisition: this stage aims at defining the mechanisms needed to provide the context. These include:
    1. Installing physical sensors and utilizing virtual sensors
    2. Knowing exactly what type of data these sensors provide
    3. Knowing exactly the way of communication with theses sensors through APIs or special software to be developed
    4. Communicating with sensors exactly as defined in the previous step in order to query and register the interest of being notified in case of changes
    5. Storing context
    6. Interpreting context

- Provisioning: this stage aims at providing methods to make the context information accessible to applications

- Reception: this stage aims at acquiring context, and can be divided into the following steps:
  1. Knowing where the needed sensors are and how to communicate with them
  2. Communicating with them
  3. Interpreting the received context to make it more useful for the application, if possible

- Action: this stage aims at adapting the application according to received context, this can be done by following the next steps:
  1. Analyzing the context where it is perceived as an independent variable or by combining it with other information collected momentarily or from past communication.
  2. Selecting the most appropriate context-aware behavior
  3. Perform the behavior

3.4 Difficulties confronting the development and maintenance of context-aware applications

Suggesting a context-awareness architecture that various applications could benefit from is a demanding task that needs a deep analysis. As part of this analysis, we elaborate the difficulties that confront the developing and maintaining of context-aware applications.

According to Fortier et al. (2007), the main difficulties facing the development and maintenance of context-aware applications can be summarized as follows:

1. Context-aware systems merge knowledge from different areas of computer science to reach the final application; this creates a need of the presence of an integrating platform.
2. The devices used to acquire context information are non-traditional and distributed; this leads to the requirement of having context information being abstracted and interpreted to be used by different applications.
3. Abstracting context should include more than the process of changing representation.
4. The final stage and the aim of acquiring context which is the adaptation to context is not an easy process.

For the sake of completion of this part, note that the activity (situation) of interest in any scenario has a set of characteristics discussed in the literature:

- Not all of its aspects can be known.
- Not all known aspects can be captured.

- Not all aspects are relevant to establish a shared situation. And,
- There are conditional dependencies between some of the aspects which can be relevant to reason about the situation.

These characteristics should be taken into consideration while developing context-aware web applications.

3.5 Databases

Looking towards creating a context-aware architecture, we must provide a way of representing and dealing with context. Object oriented approach proves its usefulness in the field of context data modeling (context modeling) as we saw in section (2.4.2) where we provided some projects using this approach to model context.

Context models are there to give an abstract description of parts of the real world that the system needs to react on. The task of a context model can be viewed as a bridge between the physical world and the context-aware (or sentient) applications.

Using a relational database means that we can use a variety of SQL commands to help reason about context. We are interested in a type of databases called Object-Relational Database, which is considered to be an improvement to the relational database. In the next sub-sections we present an overview of the object oriented concepts, and we also discuss Oracle as an object oriented database, and conclude by explaining the concept of type inheritance used in the database.

3.5.1 Object-Relational Thinking

Object relational thinking is considered to be an improvement over traditional relational databases. The object-relational paradigm has many advantages that simple relational databases do not have and that can benefit our implementation. There are two important advantages to mention:

- The ability to build systems faster.
- The ability to build and maintain systems for less cost.

These advantages emerge from the hierarchical nature of this paradigm, in addition to the reusability feature. Using an object-relational approach will provide us with these advantages. Each time a supertype or subtype is modeled; object-oriented thinking is being used.

All object oriented context modeling approaches has an intention to employ the main benefits of any object oriented approach - namely encapsulation and reusability – to cover parts of the problems arising from the dynamics of the context in ubiquitous environments. Context processing details are encapsulated on an object level in order to be hidden to other components, and the access to the contextual information is provided through specified interfaces only (Strang and Linnhoff-Popien, 2004).

We have adopted the object oriented approach done in the Hydrogen project (Hofer, et al., 2002) with a modification in the hierarchy where there is no remote context due to the fact we stated earlier as each node is just responsible for its own context. If the model we have depicted should be used in an environment where sharing of context between various nodes aside from the web application is preferable, then the same hierarchy would be satisfactory with the addition of remote context object to match its counterpart in the Hydrogen project.

The hydrogen project contains a Java object called *ContextServer* which can be accessed through a special port and waits for commands issued by adaptors and applications. This of course differs from what we are suggesting as we push toward the usage of Web Services to disseminate information about context to trusted parties.

The advantages of Objects can be summarized as follows:

- The encapsulation of operations along with data: in addition to including data as tables, objects can also have the ability to perform operations that are most likely related to the data in the object.

- The representation of Part-Whole Relationships: An object can have other objects as attributes, and the attribute objects can have their own object attributes too. This makes it possible to avoid multiple tables with primary key-foreign key relationships.

- The efficiency: the efficiency comes from the fact that object types and their methods are stored along with the data in the database; this means that there is no need for the various developers pertaining to different applications to recreate the structures and methods again and again, hence ensuring a kind of consistent standards that all developers are committed to use. The efficiency is also empowered by being able to fetch and manipulate a set of related objects as a single unit, i.e. requesting to fetch an object from the server can retrieve other objects that are connected to it.

3.5.2 Oracle as an Object Relational Database

We have chosen Oracle Database which is considered to be an Object Relational Database (ORDB) as it supports the well-known relational database design abilities as well as supporting the capabilities of Object Orientation.

Oracle provides what are known as “object types” which are user-defined types that make it possible to model whatever entities, such as various context types, as objects in the database. These object types can be created from any built-in database types and any previously created object types. Oracle Database stores metadata for user-defined types in a schema that is available to SQL, PL/SQL, Java, and other languages. Object types and related object-oriented features provide higher-level ways to organize and access data in the database. Underneath the object layer, data is still stored in columns and tables, but the programmer can work with the data in terms of the entities that make the data meaningful (Loney, 2009). Instead of thinking in terms of columns and tables when we query the database, we can simply select entities that we have created, such as the location context type.

Oracle Database implements the object-type model as an extension of the relational model, while continuing to support standard relational database functionality, such as queries, fast commits, backup and recovery, scalable connectivity, row-level locking, read consistency, and more.

Object types serve as blueprints or templates that define both structure and behavior. They are database schema objects, subject to the same kinds of administrative control as other schema objects. Application code can retrieve and manipulate these objects.

Object types are composed of attributes and methods:

- Attributes hold the data about an object. Attributes have declared data types which can, in turn, be other object types.

- Methods are procedures or functions that applications can use to perform operations on the attributes of the object type. Methods are optional. They define the behavior of objects of that type.

3.5.3 Type Inheritance

Type inheritance gives the ability to create a type hierarchy; that is a set of successive levels of increasingly specialized subtypes (child object types) that derive from a common ancestor object type called a supertype. Inheritance is the mechanism that connects subtypes in a hierarchy to their supertypes. The features of the parent object type are given to the derived subtypes through inheritance, and those subtypes should extend the parent type definition by adding new attributes or methods, or redefining methods inherited from the parent. Eventually, the resulting type hierarchy provides a higher level of abstraction for managing the complexity of an application model (Loney, 2009).

One of the important features of type inheritance is that inheritance link remains alive. This means that subtypes automatically acquire any changes made to these attributes or methods in the parent; i.e. any attributes or methods updated in a supertype are updated in subtypes as well.

Power and complexity are the two main characteristics of the inheritance relationship between a supertype and its subtypes. Power is being clearly seen from the ability to change a method in a supertype and having the change take effect in all the subtypes

downstream just by recompiling. But this same capability means that you have to consider whether or not you want to allow a type to be specialized or a method to be redefined. Similarly, for a table or column to be able to contain any type in a hierarchy is also powerful, but we, the analysts or the programmers, must decide whether or not to allow this in a particular case. Also, we may need to constrain DML statements and queries so that they pick out just the range of types that we want from the type hierarchy.

CHAPTER FOUR

CONTEXT AWARENESS MODEL
AND ANALYSIS

4.1 General View

This chapter introduces the concepts that collectively make up our models.

Based on the design process discussed in section (3.3), we provide the following two models:

- The first one models the process performed at the node initiating the communication with the web application; this includes the phases of specification, acquisition, and provisioning.
- The second one models what happens at the web application side; this includes reception and action phases.

Creating two models as proposed above instead of one; is due to our vision of the necessity of the separation of the core application logic from the logic of the various processes that aim to detect, refine, and reason about context. This separation of base behavior of the application from context related processes should improve modularity. We follow a distributed approach, where every web-enabled node (Laptop, Desktop, PDA, Smart Phones, etc…) is responsible for its own context: detection, quality and refinement, analysis, reasoning, and provisioning. The web application is just asking about certain context aspects within a pre-determined level, and the communicating node has the responsibility to provide the application with that information. This may mean that the node has some kind of autonomous behavior where it may respond with different values according to the application trustiness for example. This could be considered a kind of *agency*.

Since the proposed model follows a distributed architecture where there is no collective information existing on a centralized location, we have to create a

mechanism that informs different applications which services (Contexts) are available at this node, i.e. we can depend on this as a substitute to publishing. This responsibility is given to a class that will be discussed later and implemented on the next chapter.

In our view, separation means that each communicating party has its own issues to concern about.

The node that accesses the application has concerns of:

- what context data to collect via whatever mechanisms available,
- what analytical processes must be performed to benefit the purpose of usage at the node
- what context information to give to requesting applications and at what level (regardless of the level of context requested)
- how to utilize the resources of the node efficiently to the benefit of context management process and to other routine purposes

The node that represents the application has also its own set of issues to take care of, these may include:

- What services/information the application provides (the core application logic)
- How to take the context of usage into consideration while providing these services
- In what ways and in what parts of the application the context may play a helping role to the user

Following we present the layers of our model. Note that when going up in the hierarchy, you are moving more and more into software.

4.2 The model suggested at the nodes initiating communication with the web application

The layers of the model suggested at the node are:

- Context Catching Layer
- Context Storage Layer
- Context Processing Layer
- Context Distribution Layer

4.2.1 Context Catching Layer

This is the layer that mostly deals with hardware. Here, hardware devices are heavily employed to collect (acquire) various contextual information of the communicating node's environment. This type of sensing where hardware devices are used to collect context is called *hard sensing*. A quick instance of these devices is sensors. It is also possible to employ software mechanisms to collect data (*soft sensing*) concerning the user interaction with the device's programs and his/her interaction with the Internet.

According to Prekop and Burnett (2003) and Gustavsen (2002) and Hofer et al. (2002), there are two classifications related to these two types of sensing:

- Soft sensing refers to the logical (internal) dimension of context which is mostly determined by the user or captured by analyzing user interactions,
- Hard sensing refers to the physical (external) dimension which can be measured by hardware

It is apparent then that context information can be gathered from a variety of sources, for example the weather conditions can be acquired by a weather station installed in the building and accessed through a sensor network (Hardware) or by consuming a web service that provides such a service (Software).

In location-aware systems, position data can be gathered through GPS, mobile phone towers, badge proximity detectors, etc.

Sensing could be categorized also according to the type of sensors used, which are: physical, virtual, and logical sensors (Hofer et al., 2002).

- Physical sensors collect data that can be sensed physically like sensing temperature for example,

- Virtual sensors dives more into software like motion detection applications and consulting Outlook calendar entries,
- Logical sensing mixes the two previous sensing categories to infer the current activity of the user for example.

In this way, logical sensing needs elaboration and analysis of physical context.

The output of this layer is *raw data* that is being handled to the layer above (the Storage layer).

As there are different technologies and a variety of ways to acquire context information, the model should enhance the requirement of making the context loosely-coupled from the sensing mechanisms used.

We suggest the presence of a Universalization Sub-Layer which transmits (provides) the context data in a universal way. This way, we can connect this layer to the layer above through well-known interfaces. Doing this also enhances the modularity of the architecture by making it possible to use any kind of technology in the next layer as long as it knows how to deal with the interfaces of context providers.

4.2.2 Context Storage Layer

This layer represents the usage of a permanent memory mechanism that influences the processing done by the above layer, especially those concerning statistical analysis and historical references where context history may be used to establish trends, predict future context values, and to implement intelligent learning algorithms that help to make applications more able to adapt intelligently; to mention a few examples

of context history usage. We consider this layer to be of higher importance and we claim that its introduction in the model is also becoming more and more practically feasible due to rapid advancements in storage technology. One of the properties of this layer is the introduction of it at this position, not at a higher position like most models do.

All contextual data that are being collected (the previous layer) are fed into this layer. In addition, domain knowledge and inference rules are stored in the database. We view this layer as the kitchen that will always be used to prepare the recipes of the following layer. Querying context data stored in the database could be done through the Structured Query Language (SQL) that provides the ability to read from and write to the database at a high level of abstraction. Responsibilities of this layer encompass in addition to storing context data, maintaining the integrity of data, and the efficient utilization of storage.

4.2.3 Context Processing Layer

This is the layer that gives the taste of the whole process. It pulls instant and old contextual data from the layer below (we like to call it cupel) and does some processing (cupellation) in order to provide the required *knowledge* that the application will depend on to adapt its behavior. Note that some processing activities may also write some results back into the Context Storage Layer. Context here is being prepared into a set of levels determined according to the context type; i.e. levels help to provide coarse grained and fine grained context. Since most consumers are interested in already interpreted and aggregated information than raw data; technical data gathered and stored in the previous layer are being analyzed and reasoned about

in a way that makes context more readable and beneficial to context-aware applications. Levels mentioned above include raw data and processed context. We recommend the approach of having a context knowledge base and an inference engine that uses various inference rules and information gathered from sensors that are saved in the database to deduce the leveled context. The inference engine follows rules in the database and applies the well known forward chaining technique, for example, in order to create the specified level of abstraction needed and to help the application make the proper decision of adaptation according to the application's conditional rules.

4.2.4 Context Distribution Layer

This layer has the responsibility of delivering context to the interested requestors. At this layer there is a kind of protecting agent that will decide whether it is safe to give correct information to the requestor. Also, we put a requirement in this layer which states the necessity of providing context in a universal format that should be understood regardless of the communicating parties' (providers and consumers) hardware and software specifications. This requirement's importance is clearly apparent; it will increase the usability of the model and the applications conforming to the model, since it overcomes the heterogeneous context data sources issue.

This layer contains a Context Protection Sub-Layer, Context Publisher Sub-Layer, and Context Universalization Sub-Layer.

- Context Protection Sub-Layer

The protective shield is strongly needed when we are suggesting that there will be many distributed web-enabled devices having user profiles and other sensitive

information and which may deal with malicious applications. It will be the firewall that must be dealt with at first to take the contextual information requested. This firewall would consult a set of rules that may include user ownership rules to determine the safest action to the node. User ownership rules state for example that this kind of context information could be given to the application satisfying some criteria. So, the context protection sub-layer resembles the function of a filter that queries the database and allows only a subset of context information to be sent to each type of application according to predefined rules, (i.e. every node defines a set of constraints to determine when it is allowed to give certain aspects of context to the requesting application).

- Context Universalization Sub-Layer

Context Universalization Sub-Layer has the responsibility of converting the contextual information required at the specific level into a format that can be interpreted by all parts of the communication running whatever software on whatever hardware architecture.

Having each framework present today with its own format to describe context and with its own communications mechanisms makes it difficult, if not impossible, to look forward an open context framework. Standardized formats and protocols, on the other hand, make it more realizable and possible to achieve this goal and bring the advantage of concentrating more on the product or service being developed rather than on the communication and the type of node being communicated. According to current software technologies, *Web Services* could achieve the task and make context aware services more interoperable. Interoperability is important since it enables developers to use Web services without thinking about which programming language or operating system the services are hosted on.

- Context Publisher Sub-Layer

This layer also contains a sub-layer which we called a Context Publisher. It has the responsibility of advertising the services of the node, i.e. which contexts are provided by the node. It does this work after consulting the firewall.

4.2.5 Model Discussion

The new idea in this model is the presentation of the storage layer at that position, in addition to combining multiple layers presented in other layered models that mainly aim to give multiple levels of abstraction to same contextual data. By using a database to represent and store contextual data, we achieve a bunch of advantages that cannot be obtained by using the normal files stored in the file system. For example cookies, which are text data stored into the user's device, could be used harmfully to track user's browsing activities. This is considered to be a major privacy concern. In our model, the presence of a database should make it difficult for hackers to gain access to the contextual information (including preferences and other private data) since the access to the database needs more authentication and knowledge of other parameters of access. In addition, knowledge of the hierarchy of the database itself is needed whereas cookies put its data in files in the operating system itself.

Note that each of the model layers is considered to be a servant to the layer above; each one depends on the methods provided by its lower neighbor to accomplish the tasks of its own. So there is a great dependency on those methods but with assertion on the requirement of loosely coupling in order to make it possible to use different technologies available while keeping consistent interfaces. Also the model asserts on the requirement of providing context aspects in a standard format to overcome the heterogeneity difficulties, plus the introduction of the firewall at the top of the hierarchy. Figure (4.2.5.1) shows our model.

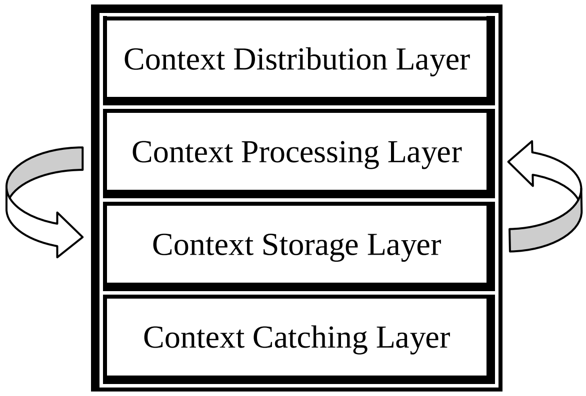


Figure 4.2.5.1: modeling what should happen on the node

This model should be realized and implemented by each web-enabled (has an IP) node. This means that we are talking about a distributed approach where sensing, context storage, and context processing are all done remotely at the node and the application is just asking the context distribution layer of the model to provide it with the needed final contextual information; i.e. there is a separation between detecting and processing context on one side and using context on the other side and this should improve modularity and reusability of systems. Making each node responsible for the control and management of its own context can be argued to be a reasonable decision. As we have mentioned above technological advancements in storage is on our side.

Also processing technology is proving superiority and thankfully making small devices capable of handling the processing level we anticipate on the context data stored. These advancements, of course, apply to both small mobile devices as well as to PCs and other comparatively big computers. Although we follow a distributed approach, we do not recommend using a central location for storage and processing due to the single point of failure problem.

As nodes may have different capabilities; a publisher object must be there to allow the application to know which sensors are attached to the node and therefore to determine

which parts of the application may be adapted. This is also a variation to the peer-to-peer systems which do not provide a discovery mechanism.

4.3 The model suggested at the web application

From the application viewpoint (i.e. application-centered viewpoint) we will model the process that the application should do in order to complete the context-awareness cycle to its end. We assume that the application is being developed using the Model-View-Controller (MVC) design pattern which leads to a separation of application code into the three mentioned layers: Model, View, and Controller.

The following is an overview of each layer as discussed by (Koletzke and Mills, 2007):

- Model: this layer defines and validates the data used by the application. It also communicates data to and from the database. The next two layers can communicate directly with this layer; the View layer is notified of any changes happening to data and the Controller layer send requests to it to update data in the application.
- View: it represents the user interface which displays data from the previous layer. It receives requests from the Controller layer to display a specific page (View). It also receives various user events.
- Controller: it determines page flow. This means that it determines which page (View) will appear according to some user event or based on specific data.

Figure (4.3.1) in the next page shows the interactions between the Model, View, and Controller of the MVC design pattern.

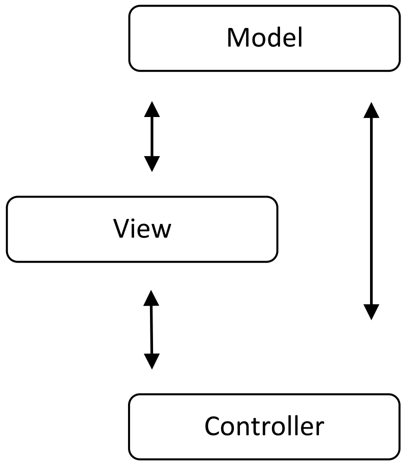


Figure 4.3.1: MVC design pattern

The basic elements of interest are as follows:

- **Database**: we argue that the adoption of a new metadata which contains besides the normal metadata attributes of data type and length, those attributes that tells the database management system which data items are context sensitive. This is done by linking each data item in a table to a contextual space. This adoption should lead to a new generation of applications that when based on this kind of databases to be more context aware. So, the new set of metadata may include: data_type, length, linked_to_context, etc…

- **Interface**: the framework used to develop the web application should give the facilities required to define which user interface item should be sensitive to context, to which type of context, and what behavior (rules) would be followed based on each type values or range of values.

- **At the top level**: there should be a handful of context consumers interacting with context providers at the other end of connection. These context consumers have the responsibility of negotiating context providers in order to grasp the context information needed at the specified level. This is considered the first step. The second one is to distribute the context information to all interested user interface items and also to interact with the model which will consult its metadata to filter and adapt the data presented by matching its metadata containing each data field existing in a table and associated with some context space to the context currently available through the various context consumers.

Figure (4.3.2) shows the objects involved in the process of context awareness at the web application node.

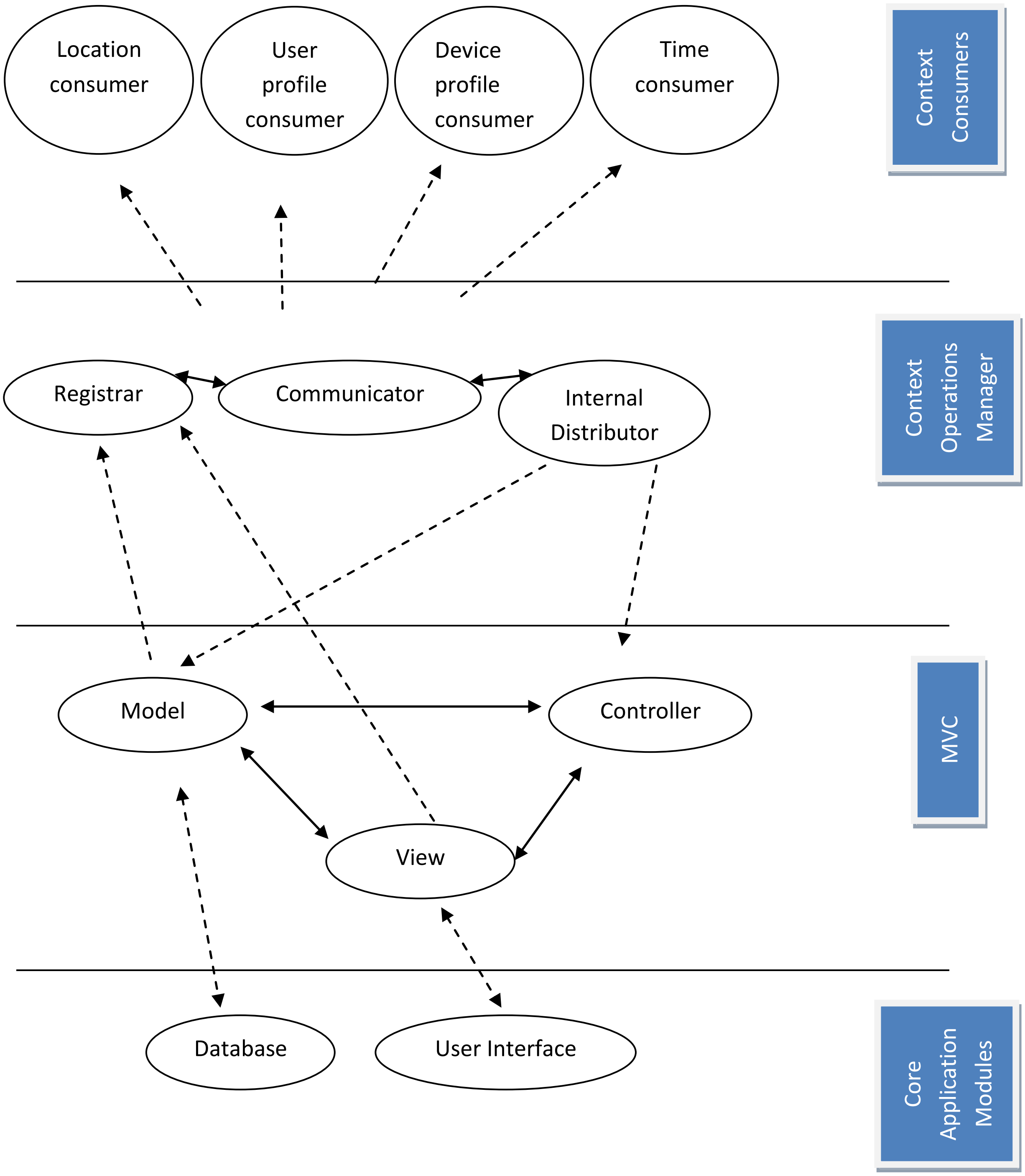


Figure 4.3.2: the web application components involved in the context awareness process

The following is an explanation of each of the components presented in figure (4.3.2).

- **Core Application Modules**: these include the primary functionalities presented by the web application and included and represented by the database, user interface, and the variety of services dealing with database and outputting to or taking input from the user interface.

- **MVC**: on top of the database and the user interface come the MVC objects which act as a mediator that talks with the upper layer on behalf of the lower layer and inherits the features and benefits talked about two pages ago.

- **Context Operation Manager**: moving bottom up, we are going more and more into being context aware. We view this class to have the main tasks of contextualizing the content and behavior of the web application. We divided its responsibilities into the following:
  - ❖ **Registrar**. The Registrar object is being contacted by the model and the view on behalf of database and user interface. Its job is simply to record that these database items are interested in this contextual information (data – content – adaptation) and that these user interface items have the potential to be adapted based on the following contextual information, etc…
  - ❖ **Communicator**. The Communicator, having channel with the upper layer, is being told by the Registrar object that this web application (database and interface) is demanding to have what types of context from the node at the other end. Armed with that information about the required types, the Communicator contacts the specified consumers –

according to a list specifying what type of context could be brought by what consumer – and waiting for their quick reply before timing out after a predetermined time in which case it will return a default value for each context type not being brought on time.

- **Internal Distributor**. When the Communicator finishes its job, the contextual information would be available to be delivered to those parts of the database and user interface that registered their sensitivity to some context.

- **Context Consumers**: for each context type there should be a context consumer. Context Consumers at the web application server node would call their counterparts, Context Providers, at the Context Distribution Layer discussed earlier. Negotiations take place to bring the context required at all available levels that could be agreed upon between the web application and the client node.

  Some types of context information could be cached to overcome any performance issues emerging from continuous demanding of the same context information by various database and user interface items.

The purpose of our models is to make the web application, which is created cooperatively by the analyst and the programmer, able to understand the activities (situations) and understand the preferences of the user when he/she is engaged in various activities.

CHAPTER FIVE

IMPLEMENTATION

AND

SCENARIOS OF USAGE

5.1 Implementation Overview

In this chapter we provide a suggested implementation of the theoretical basis recommended and analyzed in the previous chapter.

As a quick review of the models: we have two layered models representing what has to be performed on the client side and on the application server side. The one on the client side has four layers: Context Catching Layer, Context Storage Layer, Context Processing Layer, and Context Distribution Layer. The first and the third layers are out of the scope of this thesis; the first one deals with sensors and the third one is discussing the operations that could be performed on contextual data.

To meet the requirements discussed in the previous section and to prove the feasibility of the presented suggestions, we are going to provide implementations for the second layer (the database with an object oriented approach to represent contextual data) and the fourth layer (with its mentioned sub-layers). We presented these here, and demonstrated its usability by usage scenarios.

Our prototype is performed using Java programming language which follows the required Object Oriented Concepts that we recommend since Java has been object oriented since its inception. Our implementation comes in two flavors: Interfaces and Classes that implement these interfaces.

The implementation is given in two parts as was the case with modeling. The sections (5.2) through (5.5) give a suggested implementation of the model that depicts what should happen at the device which access the web application and take the responsibility of management of its own context. While section (5.6) is discussing scenarios of usage, it builds a prototype of a web application and shows what happens when the web application is accessed by a device that says my user is an instructor in

the Information Technology faculty and when the same web application is accessed by a device that says my user is a student.

5.2 The Database (Layer 2 of the first model residing at the node)

As mentioned in section (3.5.2), our database of choice is Oracle due to its object orientation capabilities.

In our implementation, we have created object tables which contain objects and nothing but objects. In these tables each row is an object (also called row object). To get a hint of what could be done, we show in the following paragraphs a variety of commands which should give a flavor of the simplicity that implementation may go through.

Figure (5.2.1) shows a command that will create a type called *CONTEXT_TYPE* which we have designed to be the parent of various context types (location, time, user, etc.). Basically, it contains the common attributes (and methods if needed) that are important and essential to all context types.

```
create or replace type CONTEXT_TYPE as object
(Capture_Time Date,
 Context_Level Number);
```

Figure 5.2.1: CONTEXT_TYPE object creation

As we have discussed in section (3.5.3), the context object should contain the common attributes of all context types the node is providing, we could for example include a CaptureTime and ContextLevel attributes that are apparently common to different context types. So, every common attribute resides at the parent object

(Context), and then the children (context types) add their own attributes. This way, we are building "objects" at the database.

On the other hand, figure (5.2.2) shows a command that will create the sub-type called *LOCATION_CONTEXT* based in a part on the previously created type which we have called *CONTEXT_TYPE*.

```
create or replace type LOCATION_CONTEXT as object
(Context Mahran.CONTEXT_TYPE,
 Location_Value VARCHAR2(222));
```

Figure 5.2.2: LOCATION_CONTEXT object creation

The other context types are implemented the same way as the *LOCATION_CONTEXT*.

Finally, figure (5.2.3) contains the command that will create what is called row objects, where objects are represented as rows in a table.

```
create table LOCATION of LOCATION_CONTEXT;
```

Figure 5.2.3: LOCATION table creation

Once we have our layer of database in place, we can access the context data stored in our Object Relational Database.

Till now, we have discussed the technologies available to support our needs in the second layer. In addition, we have shown some commands required to build the structure needed to represent contextual data. In the following sections, we will continue to show some parts of our implementation.

5.3 Interfaces

We can think of Interfaces as the generalizations that result from deep analysis of some problem. They are collections of abstract method declarations and constants, (i.e. specification of a set of methods and constants), and can be thought of as high-level abstractions of real-world objects. The reason why we have chosen to use interfaces is that interfaces are contracts that the subclasses must obey. Interfaces give us the power to define which methods are mandatory in the classes that choose to implement the Interface. The analyst who defines the Interface decides exactly what behaviors a subclass must be able to perform, but not how the behaviors will be implemented—only that they will be implemented. This gives communicating nodes the flexibility of implementation and assures that the needed methods will be implemented.

Since each class that implements an interface should provide code implementation of all the interface's abstract methods, we have adopted the idea of making each interface having as less abstract methods as possible to give the developers more flexibility since failing to provide implementations for all abstract methods makes the implementer looks as an abstract class!

Using interfaces helps us towards the implementation of Separation of Concerns (SoC) principle, the one who analyzes and structures the *"Contextualizable"* interface, for instance, is just concerned with the set of functions that must be implemented by the software at the nodes in order to provide the contextual data required. On the other hand, the software contains other concerns that must be identified and implemented. Actually we can think of SoC from another angle even when looking at the interface itself; the analyst and the implementer (the programmer) have different things to take care of. The analyst states which methods must be there,

and the programmer states which algorithms and mechanisms should be followed to process the context data and provides the final results to the return variables. Hence, SoC helps to cope with complexity, in addition to achieving robustness, adaptability, maintainability, and reusability.

To test whether a run-time object implements the *"Contextualizable"* interface, we can use *"instanceof"* operator.

Figure (5.3.1) represents part of the constants that may be included in our interface.

```
package fourthlayer;

public interface Contextualizable {

    // constants
    public static final int PHYSICAL = 1;

    public static final int SYSTEM = 2;

    public static final int PERSONAL = 3;

    public static final int NETWORK = 4;
```

Figure 5.3.1: suggested interface's CONSTANTS

Figure (5.3.2) represents part of the methods we suggest to be included in an interface that all nodes must implement.

```
    //methods
    public abstract boolean isAvailable(String ctx_type, int ctx_level);

    public abstract String getContextDesc(String cxt_type, int cxt_level);
```

Figure 5.3.2: the interface's abstract methods

An explanation of each of the most important methods in the interface is presented when discussing the Publisher class.

5.4 Classes

We do provide an implementation for the fourth layer by providing the following classes:

- Universalizer
- Publisher
- Protector

Interaction between these classes is illustrated in figure (5.4.1):

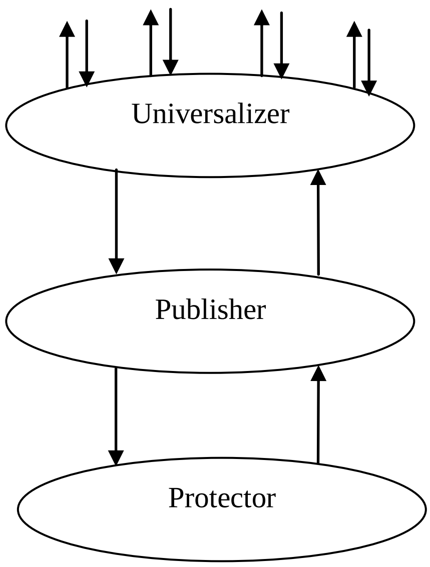


Figure 5.4.1: Fourth Layer classes and their interactions

Figure (5.4.1) suggests that the Universalizer is the part of the implementation that represents the methods available to the public. This part, although being a class as the implementation of other sub-layers of the fourth layer, is different due to having the advantage of being a web service.

The Universalizer when asked for the availability of some context at a specific level, its description, or its value, it contacts the publisher which has its own connectivity with context tables of objects in the database. Figure (5.4.2) shows a code snippet from the Universalizer class.

```
package fourthlayer;

public class Universalizer implements Contextualizable {

    Publisher p = new Publisher();

    public boolean isAvailable(String ctx_type, int ctx_level){

        return p.isAvailable(ctx_type, ctx_level);

    }
```

Figure 5.4.2: Universalizer Class – code snippet 1

Figure (5.4.3) is another code snippet from the same class.

```
public String getContextValue(String cxt_type, int cxt_level){

    return p.getContextValue(cxt_type, cxt_level);

}
```

Figure 5.4.3: Universalizer Class – code snippet 2

The methods in the previous two figures represent part of the methods available to the public. At the beginning of the Universalizer class definition is a declaration and instantiation of a Publisher object in order to utilize the methods available there. As it appears in the code snippets, and although it appears at the first glance to be a repetition of the same methods of the next class to be discussed (Publisher), it is not and there is a reason for doing it the way it has been done. We intended to code the methods which deals with the database at the Publisher class and to call them from the Universalizer class because this should give us the flexibility of changing the methods definition (signature or body or both) of the Publisher class and all we have to do, in case of changing the signature of a method, is to change the way we call it at the Universalizer, i.e. the web applications will not be affected by changes at the Publisher class since the methods at the Universalizer would have the same signature always and this is the way it should be done.

The Publisher, in order to extract the contextual data from the database, consults the protection rules to see if it is allowed to give whatever context the application wants.

Figure (5.4.4) represents a part of the "Publisher" class mentioned at the fourth layer of the model which abstracts what happens in one of the methods at the node initiating communication with the web application.

```
PreparedStatement ps = getConnection().prepareStatement("select * from context where context_type = ? and context_level = ?");
ps.setString(1,ctx_type);
ps.setInt(2,ctx_level);
ResultSet rs = ps.executeQuery();

if (rs.next()) {
  return true;
}
```

Figure 5.4.4: Publisher Class

The class contains the following functions:

- *isAvailable*: tells the web application if the current node provides a certain context type at the determined level.
- *getContextDesc*: gives a brief description of the context type provided at the level specified.
- *getContextInfo*: gives a detailed description of the context type provided at the level specified.
- *getContextValue*: returns the value of the context type sent as a parameter at the level needed.

We consider the important requirement of dynamicity when implementing the methods of the class. For example, the method "*isAvailable*" has the following signature:

- *public boolean isAvailable(String ctx_type, int ctx_level)*

The web application asks for the availability of some context type (ctx_type) at a specific level (ctx_level). If we were to implement it in another way, we'll create a special method for each context type available in the domain! , and this is of course cumbersome, as the following signatures suggest:

- *public boolean isLocationAvailable(int level)*
- *boolean isRoleAvailable(int level)*
- *public boolean isCalendarAvailable(int level)*
- *public boolean isTimeAvailable(int level)*
- etc.

but instead, the first signature of the method *isAvailable* looks more dynamic since it permits to ask for any context type whether it is contained in the database or not, and makes it possible to overcome the need to provide a separate method for each new context/level.

Implementing the recommended protection sub-layer may follow a variety of paths. Since we emphasize the use of a database; we would like to utilize it to the maximum. We put the rules of protection in tables that the Protection class consults to protect the node's context dissemination. To protect, we recommend initially using the following tables:

- A table that contains protection rules that say something like: this context at this level is allowed/not allowed to be distributed to applications residing in servers at these locations.

- A table that contains protection rules that say something like: this context at this level is allowed/not allowed to be distributed at these dates of the year (specific days or seasons) or in these hours of the day.

In addition to tables, mechanisms that knows the type/category of the web application or checks the reputation of the website by consulting a trusted source may be applied to provide the maximum protection to the node.

The first table *LOCATION_PROTECTION_RULES* should contain the valid/not valid locations of web applications in order to help provide or prevent distribution of context types at various levels.

The second table *DATETIME_PROTECTION_RULES* should contain the dates/times where it is valid/not valid to provide certain context types at specific levels. And just for the sake of simplicity, the following figure (5.4.5) could represent this table stored in the Oracle database.

COLUMN_NAME	DATA_TYPE
RULE_NO	NUMBER
CONTEXT_TYPE	VARCHAR2(111 BYTE)
CONTEXT_LEVEL	NUMBER
VALIDITY	VARCHAR2(5 BYTE)
DATETIME	VARCHAR2(20 BYTE)
DATETIME_LEVEL	NUMBER

Figure 5.4.5: *DATETIME_PROTECTION_RULES* table

Figure (5.4.6) represents a suggested implementation of the Protector sub-layer where we followed the same programming techniques as those used in the previous classes.

```
public boolean consultDateTimeProtector(String ctx_type, int ctx_level, String dtm, int dtm_level) {

try {
      PreparedStatement ps = getConnection().prepareStatement("select validity from datetime_protection_rules where context_type = ? and " +
                                                                   "context_level = ? and datetime = ? and datetime_level = ?");

      ps.setString(1, ctx_type);
      ps.setInt(2, ctx_level);

      ps.setString(3, dtm);
      ps.setInt(4, dtm_level);

      ResultSet rs = ps.executeQuery();

      if (rs.next()) {
          if (rs.getString("VALIDITY") == "YES") {
            return true;
          } else {
            return false;
          }
    } else { // there is no preventing rule; consider it allowed to give context at the specified level
        return true;
    }

  } catch (SQLException sqle) {
      sqle.printStackTrace();
      return false;
  }

}
```

Figure 5.4.6: a method in the Protector Class

As an important thing to mention, note that all previous classes and more can be stored at our database. We may use the *loadjava* utility to store a class in the Oracle database we recommended previously, and then we can access these classes by creating procedures/functions at the database itself that calls functions or procedures of the java classes.

5.5 Web Services

Hiding the complexity of any implementation of the suggested model could be realized via the usage of web services. Simply put, a web service could be thought of as an accessible application that could be used to satisfy various needs regardless of the complexity of hardware or software. Without using such a technology in the implementation of the model, we must have been otherwise doing more and more specialization of the model/implementation to suite some kind of architecture. But web services technology will fit exactly where we want it to be: the environment of heterogeneous nodes accessing a web application which on its part asks for the node's context in order to provide the application's services appropriately.

This technology is a set of related standards that allow any two applications (in our case, nodes and the web application are the parts of communication) to communicate and exchange data via the Internet.

The vision is as follows: each node, regardless of its mobility or hardware/software architecture, will have whatever implementation of the four-layer model. But regardless of the implementation, there is a part of it that we consider a must: the whole work at the end must be encapsulated and delivered as a well-known web service.

This way each node acts as an agent where agents here are web services that provide a web presence for people, places and things in the physical world (e.g., services giving the current or historical circumstances of the node, and services keeping track of peoples and object whereabouts). Other forms agents can be in: applications hosted by mobile devices that a user carries or wear (e.g., cell phones, PDAs, and headphones), and services that are provided by devices in a room (e.g., projector service, light controller and room temperature controller).

The section of web services in chapter two tells more about this very important technology.

Now to delve into the implementation, we have two options:

- Either to convert the class named *Universalizer* directly into a web service, or
- We could store the classes in the database and create special PL/SQL procedures that call the methods in the class then convert these procedures into a set of web services.

Since both options will lead us to the intended goal, we'll follow the first one. Converting the *Universalizer* class into a web service will create an XML document and some fragments of code, as the following figure (5.5.1) shows, which is put at the header of the class:

```
import javax.jws.WebService;

@WebService
```

Figure 5.5.1: converting Universalizer class into a Web Service

The deployment platform was selected to be *Java EE 1.5 with support for JAX-WS Annotations* with *SOAP 1.1 Binding* and *Document/Wrapped* as the SOAP message format which controls the structure of messages transmitted to and from the web service.

Note that each method is now present in the web service.

The following figures (5.5.2, 5.5.3, 5.5.4, and 5.5.5) show parts of the generated WSDL document for the Universalizer web service.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is Oracle JAX-WS 2.1.5. -->
<!-- Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is Oracle JAX-WS 2.1.5. -->
- <definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://fourthlayer/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://fourthlayer/"
    name="UniversalizerService">
  - <types>
    - <xsd:schema>
        <xsd:import namespace="http://fourthlayer/" schemaLocation="http://localhost:7101/Thesis-fourthlayer-context-root/UniversalizerPort?
          xsd=1" />
      </xsd:schema>
    </types>
  - <message name="isAvailable">
      <part name="parameters" element="tns:isAvailable" />
    </message>
  - <message name="isAvailableResponse">
      <part name="parameters" element="tns:isAvailableResponse" />
    </message>
  - <message name="getContextDesc">
      <part name="parameters" element="tns:getContextDesc" />
    </message>
  - <message name="getContextDescResponse">
      <part name="parameters" element="tns:getContextDescResponse" />
    </message>
  - <message name="getContextInfo">
      <part name="parameters" element="tns:getContextInfo" />
    </message>
  - <message name="getContextInfoResponse">
      <part name="parameters" element="tns:getContextInfoResponse" />
    </message>
```

Figure 5.5.2: WSDL document for the Universalizer Web Service – Part1

```
- <portType name="Universalizer">
  - <operation name="isAvailable">
      <input message="tns:isAvailable" />
      <output message="tns:isAvailableResponse" />
    </operation>
  - <operation name="getContextDesc">
      <input message="tns:getContextDesc" />
      <output message="tns:getContextDescResponse" />
    </operation>
  - <operation name="getContextInfo">
      <input message="tns:getContextInfo" />
      <output message="tns:getContextInfoResponse" />
    </operation>
```

Figure 5.5.3: WSDL document for the Universalizer Web Service – Part2

```
- <binding name="UniversalizerPortBinding" type="tns:Universalizer">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
  - <operation name="isAvailable">
      <soap:operation soapAction="" />
    - <input>
        <soap:body use="literal" />
      </input>
    - <output>
        <soap:body use="literal" />
      </output>
    </operation>
  - <operation name="getContextDesc">
      <soap:operation soapAction="" />
    - <input>
        <soap:body use="literal" />
      </input>
    - <output>
        <soap:body use="literal" />
      </output>
    </operation>
```


Figure 5.5.4: WSDL document for the Universalizer Web Service – Part3


```
- <operation name="getContextValue">
    <soap:operation soapAction="" />
  - <input>
      <soap:body use="literal" />
    </input>
  - <output>
      <soap:body use="literal" />
    </output>
  </operation>
```


Figure 5.5.5: WSDL document for the Universalizer Web Service – Part4

Note that in order to make the process of discovering and accessing the web services as effortless as possible, we have avoided the standard discovery step of the web services depicted in the well-known triangle since we assume there is an agree between the node to provide the web service and the application to call that web service with the methods' signatures provided.

5.6 Scenarios of Usage

- University of Jordan Portal (Web Site)
- Aircraft Disasters' Preventer

5.6.1 University of Jordan Portal

A portal is a web application that has the goal of integrating content from various sources, whether local or remote, into a single view consisting of separate windows.

The prototype is developed using JDeveloper which is a full-featured, integrated development environment for creating multitier Java applications. We use JDeveloper for Web Application Development and for dealing with Java in the database, in contrast to previous classes where we have used the JDBC API in order to deal with the database. Using this framework, we can create the application according to the MVC model we talked about.

Lets at first decide about the context types that we are interested in. The figure below (5.6.1.1) shows the context types that may be most important to the adaptation of UoJ portal, and then a discussion of these types and possible usages is presented.

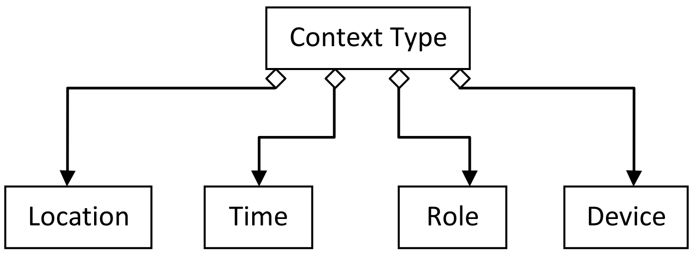


Figure 5.6.1.1: some context types of interest for the UoJ portal

- Location: gives a categorization of locations according to the portal needs, in the desired granularity: for some sections of the portal, country might be sufficient location information, for others, the faculty or center and so forth; some sections may be interested in knowing if the user is an internal or external one.

- Time refers to various types of time information that can be relevant, for example if the location indicates that this computer is in some Internet lab (Student) in the campus then the portal may be adapted according to knowing if it is the registration season, and if the computer is in an office of a faculty member then the portal may be adapted according to knowing if it is the time to enter marks.

- Role should give a categorization of users according to their role, being teachers, students, or administrator. An individual user may have several roles (being a student and an employee).

- Device gives the web application the ability to adapt according to the device type (desktop, laptop, PDA, etc) and other physical properties such as the bandwidth in such a case the adaptation could be to reduce the quality of video streaming if the bandwidth is low (Nepal and Srinivasan, 2003).

The scenario or the conversation is as follows:

The web application (the portal) is developed and deployed at *Server A*. There are other nodes (PCs) in the network that wish to access the application. For the purpose of simulation, we have used PCs (Desktops and Laptops), and we have deployed the web application and the web service on a standalone WebLogic server installed on each device (the server holding the web application and the PCs holding the web services). WebLogic is an application server that has to be configured appropriately and will work as the container for the web service and the application. Each PC will also have its own database (could be the light version of the database like those existing in the devices held by water and electricity workers who visit homes to issue receipts).

When PC 1, for instance, accesses the University of Jordan (UoJ) portal, the servlet (a type of java class) which is programmed to receive the request will extract the PC IP address in order to use it in further communications. Then the web application will see if the node is committed to give the context types of interest to the web application, so it will use the IP address of the PC and try to access the web service that should be there in the compatible nodes, and of course if there is no such a web service the application has the mechanisms of exception handling to deal with such cases and will put a default values in the variables representing the context types of interest to it. If the service is there, then further communication will take place and the chain of classes mentioned at the PC will do the job and will disseminate just those context types/levels that are allowed to be disseminated to such an application.

On the web application side, the sections of the page could, for instance, be adapted according to PCs' locations. For example, some section may be interested in

advertising to different places in UoJ, where several advertisements are being addressed to distinct locations (faculties, centers, etc…). The benefits should be apparent since contacting the PC will say to the web application at which location it sets exactly and the application will adapt accordingly by prioritizing those advertisements pertaining to the location of the node.

While implementing we tried to utilize the JDeveloper capabilities as maximum as possible, and we mixed the development by using the various wizards available in some cases (which ultimately produce code in the most efficient way affordable) and by doing the manual coding in other cases. The reason to mention such a note is to emphasize the level the rapid development of software has reached by using the current software development frameworks such as the one we used.

So if everything is almost available and web applications can be done easier than before, the issue that must be solved is how to implement our thinking and planning analysis to utilize the context of the user and mix it with the development process. Modeling should provide a way to facilitate this on the programmer.

Web services could be invoked by using wizards or programmatically. Using a wizard can facilitate the process by filling some specific web service parameters automatically and by testing the configuration. We can then modify the code generated appropriately to make it more dynamic.

Once we have the access to the web service, regardless of what approach we follow to access it, we can treat it as a class. In this manner we instantiate an object from it, and start to utilize the methods available in order to fill the variables pertaining to context types that the web application wants to adapt itself accordingly.

Figure (5.6.1.2) depicts part of the method calling and context type filling.

```
universalizerService = new UniversalizerService();
Universalizer universalizer = universalizerService.getUniversalizerPort();

if (universalizer.isAvailable("LOCATION", 3)) {
      setLocationLevel3(universalizer.getContextValue("LOCATION", 3));
} else {
      setLocationLevel3("");
    }

 if (universalizer.isAvailable("ROLE", 1)) {
        setRoleLevel1(universalizer.getContextValue("ROLE", 1));
  } else {
        setRoleLevel1("");
      }

  if (universalizer.isAvailable("TEMPERATURE", 2)) {
         setTemperatureLevel2(universalizer.getContextValue("TEMPERATURE", 2));
   } else {
         setTemperatureLevel2("");
       }
```

Figure 5.6.1.2: calling the Web Service and Filling the contextual parameters

Of course, the variables representing the context types the web application is interested in are persistent through the web site pages per session.

After having the contextual information available to the web application, it will be possible to do the adaptation. The simplest adaptation we did is the data adaptation as we mentioned earlier when we were talking about the advertisements' section. The variable called location which is available to the page were utilized directly in the query to bring those advertisements pertaining to current location of the user in addition to those public ones concerning all locations. Visual adaptation was also possible by giving the pages more elegant skin when accessed by Teachers and more lively skin when accessed by students. The organization of the page should be different as well.

So as a summary, figure (5.6.1.3) shows what happens at the servlet that receives the nodes' requests.

```
// getting the IP Address of the device
String ipAddress = getIPAddress();

// contacting the web service at the client
fillContextAspectsOfInterest(ipAddress);

// it is ok now to send the end user to the home page
response.sendRedirect("/mahran/faces/portalHomePage.jspx");
```

Figure 5.6.1.3: summary of steps done in the web application

The function *getIPAddress()* will return the IP of the node which will be sent to the procedure (*fillContextTypesOfInterest()*) that contacts the node in order to ask for the context of it. The collected contextual information would be put in variables that are accessible to all pages of the portal the whole session. Then the home page will be displayed.

It is apparent that this could be done in other ways. For instance, we could postpone the job of contacting the node till some pages are being accessed and, in the case of many pages are interested in the same context, this may mean to see if the variable is filled before contacting the node or may mean that a communication should be done each time one of these pages is being accessed.

Now we will give an overview of the organization of a sample page in the mentioned Portal. Figure (5.6.1.4) shows the page before being context aware.

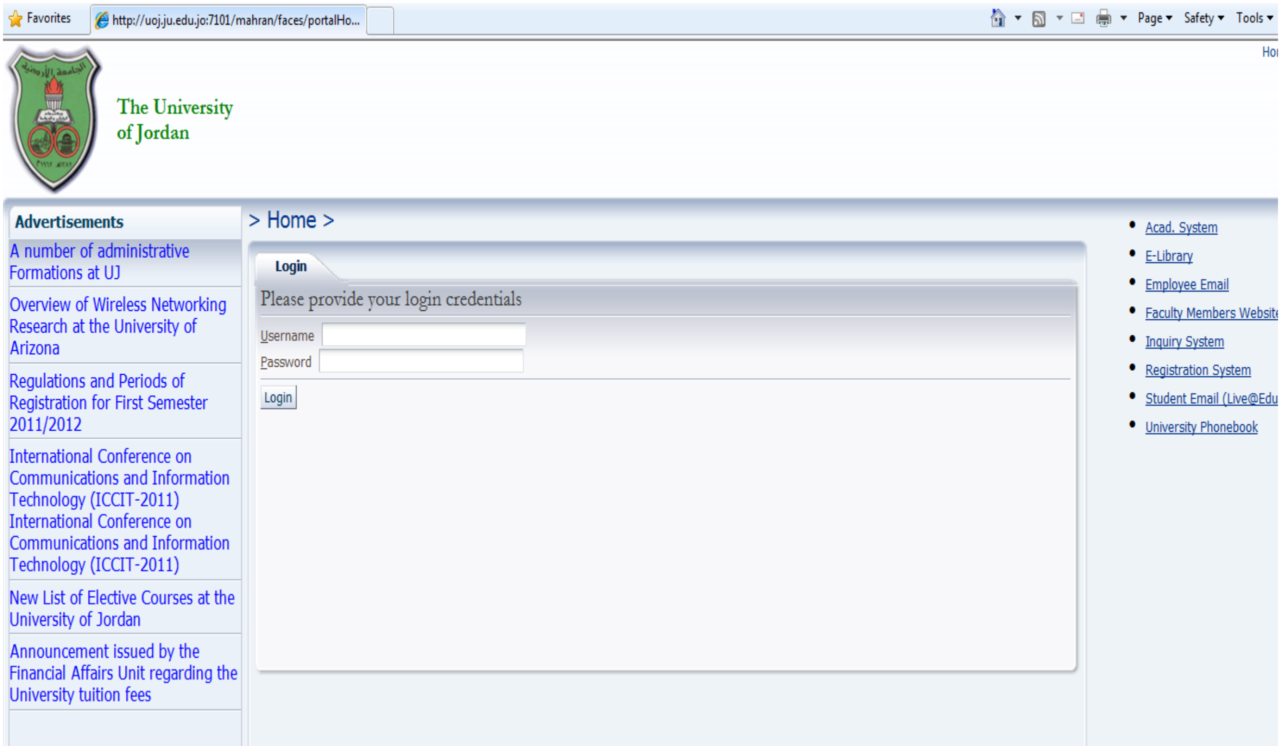


Figure 5.6.1.4: the non-context-aware version of the prototype

The page contains the following sections:

- Advertisements Section: this section contains all the advertisements of the University of Jordan usually ordered by date of publishing.
- Personal Section: it contains three tabs of information:
  - ❖ Lectures: which adapts its content to current day of week and role of the user; for example if the user is an instructor it will be filled with the lectures he teaches on this day whereas if he is a student it will be filled with lecture he must attend. In both cases, it brings its information from the Registration Unit Information System.

- ❖ Daily Schedule: it works as a personal calendar
- ❖ To do: lists the user's future activities.

- List of links Section: this section contains links to other pages or applications in the university.
- An additional section: this displays a different label and content according to the Role of the User.

The purpose of the prototype implementation is to show the apparent difference between the context-aware web application and other web application which is not context-aware.

Figure (5.6.1.5) shows the advertisement section when the user is an employee working in the Information Technology Faculty and figure (5.6.1.6) shows what happened when he was a student. This section registers its interest to adapt to Role, Time, and/or Location context aspects.



Figure 5.6.1.5: how Advertisements' Section adapts when the user Role is Employee and the Location is IT



Figure 5.6.1.6: how Advertisements' Section adapts when the user Role is User and the Location is outside UoJ Campus

As a quick reminder: the web application gets its contextual information by contacting the device used to access it and to be specific by asking the device's web service mentioned more than once in this work.

Figure (5.6.1.7) is part of the personal section which is filled after entering the credentials of the user. This section registers its interest in User, Time, and Role context aspects to adapt accordingly. The figure shows what appeared when an instructor logs in, whereas figure (5.6.1.8) shows what the same section displayed when a student logs in.

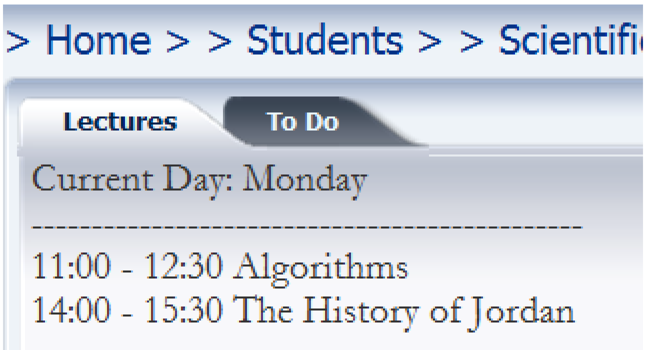


Figure 5.6.1.7: The personal section adapted according to a particular student

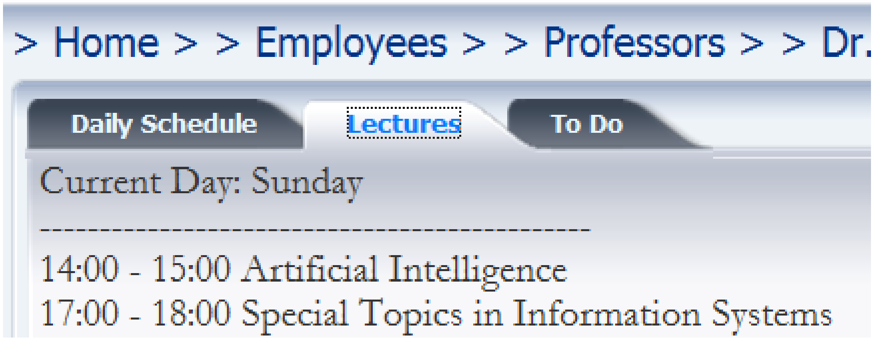


Figure 5.6.1.8: The personal section adapted according to a particular instructor

The prototype contains also a part for suggesting links to the navigator. This part is solely connected to the Role context aspect. Figure (5.6.1.9) depicts what happened in the case of an instructor while figure (5.6.1.10) shows the suggested links when the web service gives the Role as a student.

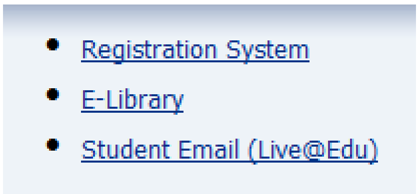


Figure 5.6.1.9: Links section adapted to Role “student”

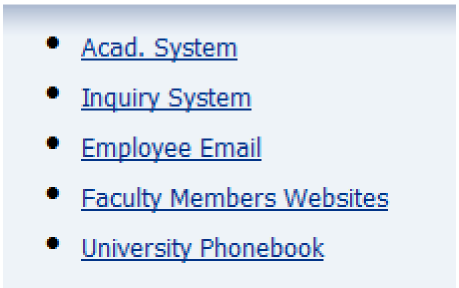


Figure 5.6.1.10: Links section adapted to Role “instructor”

The whole page with its discussed sections is seen in figure (5.6.1.11) when accessed by an instructor in the Information technology faculty, and the same page adapted to student role appears in figure (5.6.1.12).



Figure 5.6.1.11: the whole page adapted to “instructor” in “IT” on “monday”



Figure 5.6.1.12: the whole page adapted to “student” on “sunday”

Some page fragments may totally disappear or filled with information according to resources available in the proximity of the user.

Other examples of context awareness effects require the availability of more contextual information. Consider adding the specialization of the user into picture and consider adding the dimension of student's level. For example, the web application may suggest according to student's marks and the courses that have been taken in previous semesters what courses the student should register next. This way we are putting the specialization and year level into the picture to adapt content or recommend particular courses that seems to be appropriate for the student's specialization, level, and performance inferred from historical information (including his marks and might mean to see how much time spent to read a relevant material to a course on the device). The system may suggest the whole program of the semester and choose courses based on the previous contextual information in addition to choosing teaching halls close to each other. Figure 5.6.1.13 shows what happens when adapting one of the views of our prototype according to the new contextual information of specialization and year. According to student's specialization and year level and his marks in certain courses, the system may recommend what he should register next. The recommender shows the reason for recommending these courses.

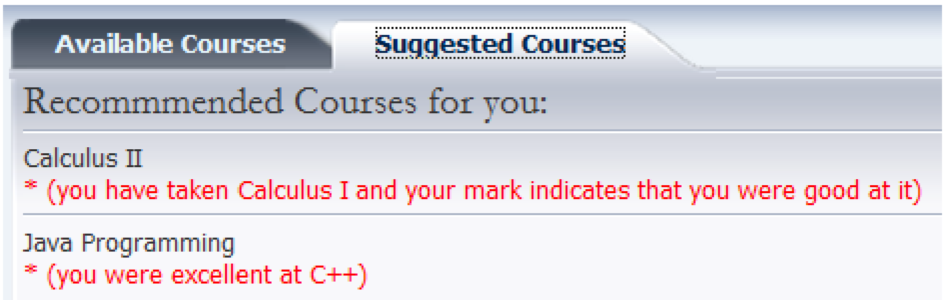


Figure 5.6.1.13: suggested courses based on specialization, level year, and performance of the student.

The recommender may also choose locations of lectures to be close to each other. Figure 5.6.1.14 contains the same content of the previous figure but chooses sections based on the location proximity and time proximity; so this example contains a bunch of contextual information.

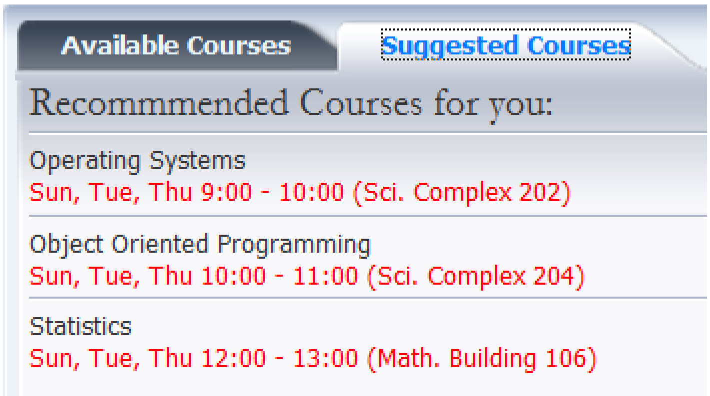


Figure 5.6.1.14: suggested courses based on specialization, level year, performance of the student, time proximity, and location proximity.

Future activities of the instructor may indicate that he will travel to some country in next few days and according to this, the web site may adapt to bring the weather conditions for example and some academic events of interest held in the destination country. To accomplish this, the specialization of the instructor and the location of the country must be available to the application as contextual information.

The academic research interests are so important in the university, and many web application portions could be adapted according to this aspect. Based on the instructor's specialization and interests, the system may introduce to him certain instructors who share with him relevant research interests.

One of the major benefits of our models is that the same contextual information could be utilized by many web applications without any need to change anything; every interested web application takes the context aspects of interest.

5.6.2 Decision Support System deployed in Airplanes

Our second scenario is an environment that is full of sensors; Airplanes. Airplanes are equipped with sensors to gather data; these valuable pieces of information is of course considered to be contextual information which will be fed into applications that should eventually advice the pilot with the procedure of steps that should be followed in order to get rid of the current confusing situation.

Different sensors are wired from various areas on the planes to the Flight Data Acquisition Unit (FDAU), which is wired to the Flight Data Recorder (FDR). FDR, as the name suggests, is a recorder of the operating data from the plane's systems; for instance, when a switch is turned on or off, that operation is recorded by the FDR. The following lists some of the sensors existing on airplanes: acceleration sensor, airspeed sensor, altitude sensor, outside temperature sensor, cabin temperature sensor, pressure sensors, engine performance sensors, and much more (HowStuffWorks.com).

What is important to us is that there is some kind of standardization that aviation community has agreed to follow; for instance, the Federal Aviation Administration (FAA) requires that commercial airlines record a minimum of eleven to twenty nine parameters, depending on the size of the aircraft. Hence, we claim that we can, on our part, agree on the naming conventions of these parameters so as to create a uniformly accepted parameter (context data) names that could be used world widely by different programmers of this kind of disaster avoidance web applications from the early stages of web development.

So we have an environment full of sensors, the raw data collected by these sensors are, as mentioned above, sent to the FDAU which sets in the middle between sensors and the FDR. This position will serve our purpose; here our model should be placed and realized. Sensors will feed a database where different statistical and analytical operations (if any) should be performed, then the context is provided through web services to any interested Information System (IS), on the ground or in the plane since most airplanes these days provide the Internet service which means that an airplane will have the web presence we request. These ISs can monitor the various contextual parameters and detect similarities with previous cases collected from various investigations for example (an implementation of CBR), and then advice the pilot with quick steps that should bring stability and may save lives of passengers or reduce the amount of loss otherwise.

At some point of time, the database will be full of records that say: the airspeed at time 14:14 was *someValue*, the temperature at the cockpit at moment 14:14 was *someValue*, the internal pressure at the passenger cabinet was *someValue*. So if something goes wrong, let's say the pilot passed away and the cockpit is full of sirens, the co-pilot may call the nearest Air Traffic Control (ATC) which has the web application that is connected to a huge database of cases and the steps to avoid disasters. The co-pilot has just to give the ATC employee the IP address and the web application will contact the agreed upon web service to ask for various contexts and eventually the ATC employee will advice the co-pilot with what he has to do.

As another variant of the same scenario, the web application may be implemented by any software vendor and installed on the airplane system in a central place. This way, we gain more time which is so precious to act against airplanes' crashes.

The following table (5.6.2.1) shows a dialogue depicting what could go on between the ATC application and the airplane:

Table 5.6.2.1: ATC application and the airplane dialogue

ATC Web Application	**The Airplane**
isAvailable("cabinet_pressure",1) Note: level 1 may mean the basic raw data measured by sensors.	*Checking security constraints*: there is no rule preventing the dissemination of this piece of information to this web application. *Checking the table which contains the types of context available at the node*: Yes (the node has this piece of information)
Since it is available can I *getContextValue("cabinet_pressure",1)* Note: the signature may be changed to indicate that I need current or historical data	*Checking security constraints*: there is no rule preventing the dissemination of this piece of information to this web application. *Checking the table which contains the values of different context types*: someValue.

CHAPTER FIVE

CONCLUSION

AND

FUTURE WORK

This chapter presents some conclusions and recommended future works.

5.1 Summary and Conclusion

This thesis introduces a new architecture for context awareness where all parties of communication cooperate to get the benefits of context awareness. The work has been accomplished and organized based on our view into two models. The first model depicts what should be done on the device accessing the web application. This layered model contains four layers: context catching, context storage, context processing, and context distribution layers. Placing context storage in this position, between the sensors which feed it and the layer responsible for different processing mechanisms, comes in contrary with what other models in the literature suggest. Putting storage before processing gives the advantage to elaborate and reason about context now and then depending on new and historical raw data. The distribution layer has a tough requirement to securely disseminate context to trusted web applications regardless of the server environment.

The second model, on the other hand, put further layers over those used while developing the web application in an effort to make it context aware. Each component, in the database or the interface, which has the capability to adapt, registers its interest in some contextual aspects. The registrar object of our suggested model carries this interest and others to the communicator object which has the responsibility to activate the relevant context consumers that on the other hand communicate with their context provider counterparts.

We have shown the applicability of our models by developing a prototype of University of Jordan Portal. The implementation has utilized a mix of current available technologies: for storage we have used Oracle 11g database due to its

support to object orientation that is needed to model context, and we have created classes in the fourth layer using Java programming language, in addition to utilizing the Web Services technology to satisfy the requirement of interoperability. The development of the web application itself was accomplished using ADF which is an integrated framework that uses Java as its programming language of choice.

And as an initiative to apply the model on another example from a distant domain we have started to study the development of a web application that may help the pilots of airplanes to avoid disasters. But this study needs more elaboration in order to make it real.

5.2 Future Work

It will be beneficial if the second model, which resides at the application server, get more elaboration since we have mostly worked on the node side. This elaboration could concentrate, among many concerns, on the ability to enhance the possibility of legacy systems to be context aware. This could happen by introducing a new layer at the top of the layered model proposed (that would be pertaining to the web application).

Some of the layers of the first model need more analysis, especially the context processing layer. The improvement should ask how exactly we can benefit from the database at the lower layer, and what kind of analysis to the context could be done that should give the results in a reasonable time.

REFERENCES

REFERENCES

Bacon, J. Bates, J. and Halls, D. (1997), Location-oriented multimedia. **IEEE Personal Communications**, 4(5): 48-57.

Baldauf, M. Dustdar, S. and Rosenberg, F. (2007), A survey on context-aware systems. **Int. J. Ad Hoc and Ubiquitous Computing**, 2(4): 263–277.

Bauer, Joseph. (2003). **Identification and Modeling of Contexts for Different Information Scenarios in Air Traffic**. Unpublished Master's Thesis. Berlin Institute of Technology, Berlin, Germany.

Beigl, M. Krohn, A. Zimmer, T. Decker, C. and Robinson, P. (2003). AwareCon: Situation Aware Context Communication, **Proceedings of UbiComp**, Berlin, 132-139.

Black boxes: **http://science.howstuffworks.com/transport/flight/modern/black-box.htm**; (last visited in Sept. 2011)

Bolchini, C. Orsi, G. Quintarelli, E. Schreiber, F. and Tanca, L. (2011). Context Modelling and Context Awareness: steps forward in the Context-ADDICT project. **IEEE**.

Bradley, N. A. and Dunlop, M. D. (2005), Towards a Multidisciplinary Model of Context to Support Context-Aware Computing. **Human–Computer Interaction**, 20, 403–446.

Brezillon, Patrick. (1999), Context in problem solving: A survey. **The Knowledge Engineering Review**, 14(1): 47-80.

Brown, P. J. (1996). The Stick-e document: A framework for creating context-aware applications, **Proceedings of EP**.

Bunt, Harry. (1997). Context and dialogue control, **Proceedings of Context**.

Cheverst, K., Mitchell, K., and Davies, N. (1999), Design of an object model for a context sensitive tourist GUIDE. **Computers and Graphics**, 23(6): 883-891.

Chen, Harry (2004), **An Intelligent Broker Architecture for Pervasive Context-Aware Systems**. Unpublished PhD dissertation, University of Maryland, USA.

Chen, H. Finin, T. and Joshi, A. (2003). Using OWL in a Pervasive Computing Broker, **Proceedings of Workshop on Ontologies in Open Agent Systems** (AAMAS 2003).

Compact Oxford English Dictionary of Current English. 2005. Third Edition. New York, NY: Oxford University Press.

Dey, A. K. Abowd, G. D. and Wood, A. (1999), CyberDesk: A Framework For Providing Self-Integrating Context-Aware Services. **Knowledge-Based Systems**, 11, 3–13.

Dey, Anind. (2001), Understanding and Using Context. **Personal and Ubiquitous Computing**, 5(1):4-7.

Dourish, Paul. (2003), What We Talk About When We Talk About Context. **Personal and Ubiquitous Computing**, 8: 19–30.

Dumke, R. Lother, M. Wille, C. and Zbrog, F. (2003). **Web Engineering**. Pearson Studium.

Fetzer, Anita. (1997). Recontextualizing Context, **Proceedings of Context Organiser workshop** at ECCS, Manchester, UK, 176-180.

Fortier, A. Canibano, N. Grigera1, J. Rossi1, G. and Gordillo, S. (2007). An Object-Oriented Approach for Context-Aware Applications, ISC'06 **Proceedings of the 14th international conference on Advances in smalltalk**, Prague, Czech Republic, 23-46.

Funk, H. B. and Miller, C. A. (1997). Context sensitive interface design, Proceedings of CONTEXT '97.

Gray, P., and Salber, D. (2001). Modelling and Using Sensed Context Information in the design of Interactive Applications, **Proceedings of 8th IFIP International Conference on Engineering for Human-Computer Interaction** (EHCI 2001), Toronto,Canada, May, 2001.

Gu, T. Wang, X. H. Pung, H. K. and Zhang, D. Q. (2004). Ontology Based Context Modeling and Reasoning using OWL, **Proceedings of the 2004 Communication Networks and Distributed Systems Modeling and Simulation Conference** (CNDS2004), San Diego, CA, USA, January, 2004.

Gunzer, Hartwig. (2002). **Introduction to Web Services**, Sales Engineer.

Halpin, Terry. (2001), **Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design**. Morgan Kaufman Publishers, San Francisco.

Held, A. Buchholz, S. and Schill, A. (2002). Modeling of context information for pervasive computing applications, **Proceedings of SCI 2002/ISAS 2002**.

Hofer, T. Schwinger, W. Pichler, M. Leonhartsberger, G. and Altmann, J. (2002). Context-awareness on mobile devices – the hydrogen approach, **Proceedings of the 36th Annual Hawaii International Conference on System Sciences**, 292–302.

Hunter, D. Watt, A. Rafter, J. Duckett, J. Ayers, D. Chase, N. Fawcett, J. Caven, T. and Patterson, B. (2007). **Beginning XML**, 4th edition, Wiley Publishing.

Jahnke, J. Bychkov, Y. Dahlem, D. and Kawasme, L (2004). Context-Aware Information Services for Health care, **Proceedings of the KI-2004 International Workshop on Modelling and Retrieval of Context**, Vol-114., CEUR.

Jancsary, J. Neubarth, F. Schreitter, S. and Trost, H. (2011). Towards a Context-Sensitive Online Newspaper. **Proceedings of the 2011 Workshop on Context-awareness in Retrieval and Recommendation**, 2-9.

Koletzke, P. and Mills D. (2007). **Oracle JDeveloper 10g for Forms and PL/SQL Developers**, USA: McGraw-Hill.

Lamming, M. and Flynn, M. (1994). Forget-me-not: intimate computing in support of human memory, **Proceedings FRIEND21 Symposium on Next Generation Human Interfaces**, 125-128.

Levandoski, J. Mokbel, M. and Khalefa, M. (2010). Caredb: A context and preference-aware location-based database system. **PVLDB**, 3(2):1529-1532.

Loney, Kevin (2009). **Oracle Database 11g The Complete Reference**, USA: McGraw-Hill.

Louis, S. J. and Shankar, A. (2004), Context learning can improve user interaction. In Du Zhang, ´Eric Gr´egoire, and Doug DeGroot, editors, IRI, 115-120. **IEEE Systems**, Man, and Cybernetics Society.

McCarthy, John. (1993). Notes on formalizing context, **Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)**, 555-562.

Michlmayr, A. Rosenberg, F. Platzer, C. Treiber, M. and Dustdar, S. (2007). Towards Recovering the Broken SOA Triangle – A Software Engineering Perspective, **Proceedings of the 2nd International Workshop on Service Oriented Software Engineering (IW-SOSWE)**.

Sathan, D. Meetoo, A. and Subramaniam, R. (2010), Context Aware Lightweight Energy Efficient Framework. **International Journal of Human and Social Sciences** 5(4):249-255.

Schilit, B. Adams N., and Want R. (1994). Context-aware computing applications, **Proceedings of the Workshop on Mobile Computing Systems and Applications**, Santa Cruz, CA, December 1994, 85-90.

Schilit, B. LaMarca, A. Borriello, G. Griswold, W. G. McDonald, D. Lazowska, E. Balachandran, A. Hong, J. and Iverson, V (2003). Challenge: ubiquitous location-aware computing and the "place lab" initiative, In WMASH '03: **Proceedings of the 1st ACM international workshop on Wireless mobile applications and services on WLAN hotspots**, New York, NY, USA, 29-35.

Schmidt, A. Beigl, M. and Gellersen, H.-W (1999), There is more to context than location. **Computers and Graphics**, 23, 6, 893–901.

Schmidt, A., and Laerhoven, K. V. (2001), How to Build Smart Appliances. **IEEE Personal Communications**, 8(4): 66-71.

Schreiber, F. Camplani, R. Fortunato, M. Marelli, M. and Rota, G. (2011). Perla: A language and middleware architecture for data management and integration in pervasive information systems. **IEEE Software Engineering**.

Strang, T. and Linnhoff-Popien C. (2004). A Context Modeling Survey. **Workshop on Advanced Context Modelling, Reasoning and Management as part of UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing**, Nottingham/England, September 2004.

Valums, Andrew (2010), Web apps vs Desktop apps, Retrieved October 2011 from **http://valums.com/web-apps/**

Wan, Kaiyu. (2009), A Brief History of Context. **IJCSI International Journal of Computer Science**, 6(2).

Wang, X. H. Zhang, D. Q. Gu, T. And Pung, H. K. (2004). Ontology Based Context Modeling and Reasoning using OWL, **In Workshop Proceedings of the 2nd IEEE Conference on Pervasive Computing and Communications (PerCom2004)**, Orlando, FL, USA, March, 2004), 18–22.

Ward, A. Jones, A. and Hopper, A. (1997), A new location technique for the active office. **IEEE Personal Communications**, 4(5), 42–47.

Webster's New World Dictionary, 1970. Second College Edition. New York And Clevland. The World Publishing Company.

Winograd, Terry (2000), Architectures for Context. **Human Computer Interaction**, 15(4), 263-322.

Zhang, Xiaohui (2011). A User's Perspective of Design for Context-Awareness, **Proceedings of the 13th international conference on Ubiquitous computing**, 531-534.

نحو نموذج لوعي السياق باستخدام خدمات الويب

إعداد
مهران متعب الزيود

المشرف
الاستاذ الدكتور نديم عبيد

الملخص

شهدت السنوات الأخيرة انتقال الكثير من التطبيقات من بيئات التشغيل التقليدية المغلقة إلى بيئات تتميز بانفتاحها على شرائح أوسع من المستخدمين الذين يستخدمون أجهزة تتميز بإمكانيتها المتنوعة. هذه التطبيقات التي يتم الولوج إليها من خلال مستعرضات الويب تقدم عادةً الكثير من المعلومات والخدمات. هاتان الحقيقتان – البيئات الأكثر انفتاحاً والتضخم في المحتوى المقدّم – يشكلان عقبة حقيقية في وجه الوصول إلى الدرجة المطلوبة من الاستفادة من تطبيقات الويب؛ حيث تقيس درجة الاستفادة مستوى رضى مزوّد هذه الخدمات ومستخدمها على حد سواء.

يمكن تحسين مستوى الاستفادة من تطبيقات الويب إذا ما تمكنت هذه التطبيقات من معرفة حالة المستخدم والظروف المحيطة به أثناء استخدامه للتطبيق؛ وبالتالي العمل على تشكيل ما يتم تقديمه على النحو الذي يناسب هذا المستخدم. تتوفر القدرة والامكانية المنشودة لتغيير هذه التطبيقات للأفضل في معرفة بيئة المستخدم وظروفه، على أن يتم تعاون طرفي الاتصال لتحقيق النتيجة المرجوة من القدرة على الحصول على ما يتعلق بالمستخدم سواء دوره وما يفضله أو ما يتعلق ببيئته المباشرة.

في محاولة منا لتحسين مستوى الاستفادة من التطبيقات المتوفرة على الويب؛ ركّزنا في هذه الرسالة على نمذجة وعي السياق. تم تقديم نموذجين يكملان بعضهما البعض وينظمان المعرفة في طبقات تهدف إلى إعطاء مطوّري تطبيقات الويب المعرفة اللازمة والتصور المطلوب لتطوير تطبيقات ويب واعية لسياق المستخدم وتتصرف بناء عليه.

إثنتان من أهم القضايا التي واجهتنا أثناء العمل على إعطاء هذه النماذج وجهاً عملياً هما: كيفية تمثيل السياق ومسألة عدم التجانس بين مختلف أطراف الاتصال الذي يميز بيئة تطبيقات الويب المنفتحة. لتذليل هاتين العقبتين قمنا باعتماد أسلوب الكائنات الموجهة لتمثيل السياق، بالإضافة إلى استغلال منهجية خدمات الويب لتوفير نوع من الترابطية على المستوى الذي يشمل كافة الأجهزة في بيئة الويب والذي من شأنه تسهيل التواصل ما بين الخادم الذي يحتوي التطبيق وتلك الأجهزة التي تلج إليه.

تم اختيار حالتي دراسة من نطاقين متباعدين: الأولى عبارة عن نموذج أولي لبوابة الكترونية مقترحة للجامعة الأردنية وقد تم تكوين هذا النموذج عملياً، والثانية هي نظام يساعد على اتخاذ القرارات في الحالات الحرجة التي قد تمر بها الطائرات. الحالة الثانية تحتاج للمزيد من الوقت للدراسة المستفيضة.